ON SIMULATION METHODOLOGY IN VEHICULAR TRAFFIC FLOW

A THESIS
Presented to
The Faculty of the Division of Graduate
Studies and Research
By
Michael Pierce Deisenroth

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in the School of Industrial and Systems Engineering

Georgia Institute of Technology
March, 1974
ON SIMULATION METHODOLOGY IN VEHICULAR TRAFFIC FLOW

Approved:

D. C. Montgomery
R. G. Heikes
R. S. Jones
F. H. Wright
D. B. Young

Date approved by Chairman: May 8, 1974
ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. D. C. Montgomery who served as my advisor throughout this research. He provided the guidance and encouragement which was necessary for successful completion of this work.

I am also grateful to the other members of the reading committee, Dr. R. G. Heikes, Dr. P. S. Jones, Dr. P. H. Wright and Dr. D. B. Young. Their suggestions and observations have greatly improved this dissertation.

Dr. R. N. Lehrer, Director of the School of Industrial and Systems Engineering, has provided leadership and support. My special thanks go to him for his help and understanding.

I also wish to thank Mrs. Betty Sims. Her excellent typing has contributed to the quality of both of my theses.

My most special thanks go to my wife Claudia for her patience, encouragement, endurance and faith. I am greatly indebted to her and our two daughters, Meg and Michele, for their sacrifice and support.

This work is dedicated to my students. May I show them the patience and understanding that others have shown me.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>vii</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>ix</td>
</tr>
<tr>
<td>Chapter</td>
<td>1</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>Purpose of Research</td>
<td></td>
</tr>
<tr>
<td>Limitations and Scope</td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td></td>
</tr>
<tr>
<td>II. LITERATURE SURVEY</td>
<td>4</td>
</tr>
<tr>
<td>Simulation Situations</td>
<td></td>
</tr>
<tr>
<td>Intersection Simulation</td>
<td></td>
</tr>
<tr>
<td>Network Simulation</td>
<td></td>
</tr>
<tr>
<td>Freeway and Highway Simulation</td>
<td></td>
</tr>
<tr>
<td>Modeling Philosophies</td>
<td></td>
</tr>
<tr>
<td>Physical Representation Versus Memorandum Representation</td>
<td></td>
</tr>
<tr>
<td>Time Scanning Methodologies</td>
<td></td>
</tr>
<tr>
<td>III. DESCRIPTION OF THE MODELS</td>
<td>13</td>
</tr>
<tr>
<td>Common Features of the Models</td>
<td></td>
</tr>
<tr>
<td>The General Form of the Network</td>
<td></td>
</tr>
<tr>
<td>Pseudo-Random Number Generation</td>
<td></td>
</tr>
<tr>
<td>Vehicle Generation</td>
<td></td>
</tr>
<tr>
<td>Turning Behavior</td>
<td></td>
</tr>
<tr>
<td>Speed Distribution</td>
<td></td>
</tr>
<tr>
<td>Left Turn Gap Acceptance</td>
<td></td>
</tr>
<tr>
<td>Measures of Effectiveness</td>
<td></td>
</tr>
<tr>
<td>The Continuous Model</td>
<td></td>
</tr>
<tr>
<td>Roadway Representation</td>
<td></td>
</tr>
<tr>
<td>Vehicle Representation</td>
<td></td>
</tr>
<tr>
<td>Vehicle Movement</td>
<td></td>
</tr>
<tr>
<td>Acceleration Restriction</td>
<td></td>
</tr>
<tr>
<td>Car-Following Restriction</td>
<td></td>
</tr>
<tr>
<td>Stopping Restriction</td>
<td></td>
</tr>
</tbody>
</table>
Chapter

III. DESCRIPTION OF THE MODELS (Continued)

Turning Restriction
Measures of Effectiveness
General Model Flow
The Unit Block Model
Roadway Representation
Vehicle Representation
Vehicle Movement
  Intra-Link Movement
  Inter-Link Movement
Measures of Effectiveness
General Model Flow
The Zone Model
Roadway Representation
Vehicle Representation
Traffic Movement
  Intra-Link
  Inter-Link
Measures of Effectiveness
General Model Flow
The Next-Event Model
Roadway Representation and Traffic Movement
Vehicle Representation
Vehicle Movement and the Event Chain
Measures of Effectiveness
General Model Flow

IV. COMPARISON AND EVALUATION OF MODEL PERFORMANCE. . . . . . . . . . . . 60

Model Verification and Validation
  Model Verification
    Realism of Traffic Movement
    Pseudo-Random Number Tests
    Vehicle Generation Testing
  Model Validation
    Webster's Equation
    Results of Tests
Network Performance Comparison
  Description of Test Networks
    Network Considerations
    Selection of Signal Settings
    Volume Considerations
  Test Results
    Model Performance
    Computer Requirements
IV. COMPARISON AND EVALUATION OF MODEL PERFORMANCE

Secondary Considerations
  Scan Time Effects
  Network Preloading
  Interarrival Time Distributions

V. CONCLUSIONS AND RECOMMENDATIONS .................................. 96

Summary
Conclusions
  Effects of Model Philosophies on Performance
  Warm-up Time and Preload Considerations
  Desirable Step Size
  Interarrival Time Distribution
Recommendations
  Direction of Future Traffic Models
  Model Validation Practices
  Performance Measures

APPENDIX

A. INPUT FORMAT SPECIFICATIONS ...................................... 102
B. FORTRAN LISTING OF CONT MODEL ................................. 105
C. FORTRAN LISTING OF UB MODEL ................................. 133
D. FORTRAN LISTING OF ZONE MODEL ................................. 168
E. FORTRAN LISTING OF NEXT MODEL ................................. 191
F. LINK VOLUMES FOR CLOSED NETWORK .............................. 222
G. AUTOCORRELATION FUNCTION OUTPUT ............................... 223

BIBLIOGRAPHY OF CITED LITERATURE .................................. 240

BIBLIOGRAPHY OF SIMULATION OF TRAFFIC FLOW .................. 244

VITA ............................................................................ 260
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Intersection End Points</td>
<td>20</td>
</tr>
<tr>
<td>2.</td>
<td>Intersection Turn Points</td>
<td>26</td>
</tr>
<tr>
<td>3.</td>
<td>Distribution of Desired Speeds</td>
<td>36</td>
</tr>
<tr>
<td>4.</td>
<td>Minimum Acceptable Headway</td>
<td>37</td>
</tr>
<tr>
<td>5.</td>
<td>Possible Vehicle States of Vehicle in NEXT Model</td>
<td>54</td>
</tr>
<tr>
<td>6.</td>
<td>Turn Percentages in Validation Tests</td>
<td>65</td>
</tr>
<tr>
<td>7.</td>
<td>Average Delay per Vehicle for Validation Study</td>
<td>67</td>
</tr>
<tr>
<td>8.</td>
<td>Model Warm-up Times</td>
<td>76</td>
</tr>
<tr>
<td>9.</td>
<td>Average Delay per Vehicle at Main Intersection of Test Networks</td>
<td>77</td>
</tr>
<tr>
<td>10.</td>
<td>Analysis of Variance of Model Comparison Data</td>
<td>78</td>
</tr>
<tr>
<td>11.</td>
<td>Program Execution Time for Test Runs</td>
<td>85</td>
</tr>
<tr>
<td>12.</td>
<td>Computer Memory Requirements</td>
<td>86</td>
</tr>
<tr>
<td>13.</td>
<td>Effects of Scan Time on ZONE Model Response Predictions</td>
<td>88</td>
</tr>
<tr>
<td>14.</td>
<td>Analysis of Variance of the Effects of Scan Time on the ZONE Model</td>
<td>88</td>
</tr>
<tr>
<td>15.</td>
<td>Effects of Clock Units on NEXT Model Response Predictions</td>
<td>91</td>
</tr>
<tr>
<td>16.</td>
<td>Analysis of Variance of Clock Unit Effect on NEXT Model Response Predictions</td>
<td>91</td>
</tr>
<tr>
<td>17.</td>
<td>Effects of Interarrival Time Distribution on the NEXT Model Response Distribution</td>
<td>94</td>
</tr>
<tr>
<td>18.</td>
<td>Analysis of Variance of Effects of Interarrival Time Distribution on NEXT Model</td>
<td>95</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Illustrative Traffic Network.</td>
<td>14</td>
</tr>
<tr>
<td>2.</td>
<td>Gap Acceptance Distribution</td>
<td>19</td>
</tr>
<tr>
<td>3.</td>
<td>Link Endpoints for Two Types of Intersections</td>
<td>21</td>
</tr>
<tr>
<td>4.</td>
<td>General Flow of CONT Model</td>
<td>30</td>
</tr>
<tr>
<td>5.</td>
<td>Unit Block Representation of a Two-Lane Road Segment.</td>
<td>33</td>
</tr>
<tr>
<td>6.</td>
<td>Intersection Diagrams for the UB Model</td>
<td>35</td>
</tr>
<tr>
<td>7.</td>
<td>Intersection Maneuvers for the UB Model</td>
<td>39</td>
</tr>
<tr>
<td>8.</td>
<td>General Flow of the UB Model</td>
<td>42</td>
</tr>
<tr>
<td>9.</td>
<td>Two Two-Lane Links Divided into Zones</td>
<td>44</td>
</tr>
<tr>
<td>10.</td>
<td>Platoon Dispersion Utilizing Transfer Formula</td>
<td>48</td>
</tr>
<tr>
<td>11.</td>
<td>General Flow of the ZONE Model</td>
<td>50</td>
</tr>
<tr>
<td>12.</td>
<td>Event Sequence for NEXT Model</td>
<td>53</td>
</tr>
<tr>
<td>13.</td>
<td>General Flow of NEXT Model</td>
<td>58</td>
</tr>
<tr>
<td>14.</td>
<td>Model Verification Networks</td>
<td>62</td>
</tr>
<tr>
<td>15.</td>
<td>Comparison of Webster Equation to Predictions from Models for No Turns.</td>
<td>68</td>
</tr>
<tr>
<td>16.</td>
<td>Comparison of Webster Equation to Predictions from Models for 5% Left-5% Right Turns</td>
<td>69</td>
</tr>
<tr>
<td>17.</td>
<td>Comparison of Webster Equation to Predictions from Models for 10% Left-10% Right Turns</td>
<td>70</td>
</tr>
<tr>
<td>18.</td>
<td>Link-Node Diagrams of Test Networks</td>
<td>72</td>
</tr>
<tr>
<td>19.</td>
<td>Time-Space Diagrams for 90-Second Cycles</td>
<td>74</td>
</tr>
<tr>
<td>20.</td>
<td>Sample Autocorrelation Function and Partial Autocorrelation Function for North Approach Link</td>
<td>82</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>21.</td>
<td>Sample Autocorrelation Function and Partial Autocorrelation Function for South Approach Link</td>
<td>83</td>
</tr>
<tr>
<td>22.</td>
<td>Basic Concept of Coded Data Words</td>
<td>86</td>
</tr>
<tr>
<td>23.</td>
<td>Scan Time-Flow Interaction for ZONE Model</td>
<td>89</td>
</tr>
<tr>
<td>24.</td>
<td>Interaction Between Clock Unit and Flow for NEXT Model</td>
<td>92</td>
</tr>
</tbody>
</table>
SUMMARY

Earlier work in modeling traffic flow of the sort treated here has established that analytical models are too limited but that the general technique of stochastic event simulation is fruitful. Several traffic simulation methodologies have evolved. This research compares and contrasts the methodologies in order to provide insight into future areas of model research and development.

Four computer models, representing different strategies for simulating urban vehicular traffic flow, were programmed and tested. Webster's experimental model was used to validate each simulation model for an isolated intersection.

A series of tests were conducted to determine the significance of model type, network configuration and flow rate. A statistically significant difference in the delay predictions of the models was found. Although the time-dependent characteristics of delay predictions obtained from the various models appeared to follow the same pattern, the magnitudes of delay predicted by the models differed significantly.

Additional tests were run on selected models to investigate some secondary considerations. The scan time interval selected was found to significantly affect delay predictions. Although the results regarding model preloading were not conclusive, evidence indicated that network preloading should result in reduced computation time for large congested systems. The correlation between delays observed in different time
periods appeared to be restricted to a one-cycle lag. However, more research is warranted before results in this area can be used to indicate simulation running time.
CHAPTER I

INTRODUCTION

The design or improvement of traffic control systems depends on a model that simulates the relevant features of the vehicular flow and the controlling mechanism. Earlier work in modeling traffic flow of the sort treated here has established that analytical models are too limited but that the general technique of stochastic event simulation is fruitful. Several traffic simulation methodologies have evolved. Their piecemeal development and reporting in the literature tend to make their distinctions and interrelationships obscure, and it is this situation that creates the need for the research reported in the present thesis.

Purpose of Research

The purpose of this research is to compare and contrast existing methodologies in the simulation of traffic flow. It is hoped that these studies will provide a means of evaluating the relative merits of these different strategies. By identifying the strengths and weaknesses of the different techniques, insight into future areas of model research and development should be realized.

Limitations and Scope

The research is limited to models of traffic flow in urban vehicular networks with specific emphasis on models used to evaluate traffic control alternatives. Specifically excluded from this research
is the work done in the areas of open road simulation and freeway
design. Although the fundamental modeling techniques are similar, dif­
ferent system configurations and vehicular behavior characteristics are
considered. (Freeway and open road models typically have longer roadway
segments and special intersection geometry where passing and merging
movements are considered.)

Four different models, representing basic modeling strategies,
are compared with respect to three different traffic situations--the
isolated intersection, an urban arterial flow and a medium-sized closed
network. Each situation is further stratified to test the effects of
traffic volumes on model performance. Light, medium and heavy service
volumes are considered for each system configuration.

Objectives
The primary objective of this research is the identification,
isolation, comparison and evaluation of different simulation methodolo­
gies as applied to traffic flow. Consideration is given to model size,
computation speed, flexibility and response prediction. Both dynamic
and static behavior of the models are compared.

As a secondary objective, three tactical problems of traffic
simulation were investigated. First, different models and situations
are tested to evaluate model initialization and warm-up. Preloading of
the roadway is investigated for two of the models considered. Next the
time-scan interval of two models is varied to test the sensitivity of
simulation response and computation time to this parameter. Finally,
the dynamic output of the models is investigated to analyze its effect on total simulation running time.
CHAPTER II

LITERATURE SURVEY

In the last two decades, digital simulation of vehicular traffic flow has attracted considerable interest among traffic flow theorists and practitioners. A review of all the literature that has been published in this area is beyond the scope of this study. Appended to this thesis is a bibliography of relevant literature which represents the majority of available literature in this area. This review will concentrate on outlining the general areas of traffic simulation and the different simulation approaches which have been employed.

Simulation Situations

Traffic flow simulation can be divided into three basic areas: intersection models, network simulation and freeway or highway models.

Intersection Simulation

Considerable effort has been devoted since 1956 to the simulation of the individual intersection. The work of Goode, Pollmar and Wright (18)* represents the pioneering contribution. A single four-legged intersection with a fixed-time traffic signal was modeled. Turning movements were permitted and left-turning vehicles checked the approaching traffic for an acceptable gap before turning. The roadway was

*Numbers enclosed in parentheses indicate references listed in the Bibliography of Cited Literature.
represented by a string of binary digits indicating the presence or absence of a vehicle. Every quarter second the vehicles were examined to see if they could be advanced.

Next, two separate studies of intersection simulation were performed by Benhard (2) and Lewis (27) in 1959. These studies dealt with the simulation of an intersection of two two-lane streets with actuated signal control. The models were greatly simplified, however, in that turning and passing were prohibited.

Lewis continued his work in intersection simulation with the development of a second model (28). This model was developed to determine volume warrants for intersection control. Two types of intersection control were studied: the two-way stop sign and the semi-actuated signal. The model permitted both turning and passing maneuvers and parking was accommodated on the minor street. The individual approaches were represented by a one-dimensional coordinate system and vehicles were permitted to assume any desired position along the approach. Extensive car-following rules were developed for use with the model. A scan time of one second was employed.

Kell (23,24) constructed models to study intersection delay for two-way stop, fixed-time and vehicle-actuated intersections. The traffic volumes were varied and the resulting delays were compared for each control method. Additional runs were made at selected volumes to determine the effect of turning maneuvers on intersection delay. By concentrating on time relationships among the vehicles, an event scan program was developed which boasted a 7000:1 ratio of real time to simulation time.
Gerlough and Wagner (16) developed a model to study traffic performance at an isolated intersection. A continuous spacing representation similar to Lewis's was adopted. Many driver characteristics were represented by probability distributions. A vehicle's following behavior was dictated by a reciprocal spacing model. In this model acceleration (deceleration) was proportional to relative speed and inversely proportional to spacing. A scan time between 0.25 and 0.5 seconds was permitted.

After having performed extensive field work to obtain relevant distributions regarding the models, Thomasson (36) and Wright (43) simulated stop sign controlled intersections, two-way and four-way, respectively. Specific emphasis was placed on driver reaction time and lag and gap acceptance.

Apparently the latest published work in the area of intersection simulation as of early 1974 was reported by Barnes (1) in 1971. The primary goal of his research was to formulate general models of seven basic types of intersections ranging from a four-way stop model to a four-lane, two-lane signalized model with two turn lanes. These models were written in GPSS, a general purpose simulation language.

Network Simulation

As soon as initial efforts were completed on simple intersection simulations, research workers began to consider the problems of simulating a portion of an urban transportation network. Goode and True (19) combined four single-intersection models to form the first network model. An oscilloscope was used to display the movement of vehicles through the network.
In 1961, Stark (35) developed a model for a nine-block section on 13th Street, N.W., Washington, D.C. The model represented a combination of signal-controlled and stop-sign-controlled intersections. Additionally, vehicles were allowed to pass as well as accelerate and decelerate. A total of 37 main routines, subroutines and table look-up routines were included in the model. The roadway was divided into 12-foot unit blocks. Positioning could be specified to the nearest 1/100th of a block. The roadway was updated every quarter of a second.

Katz (22) and Gerlough and Wagner (13) reported the initial development of TRANS, the first general network simulator, in 1963. TRANS was written to evaluate the effects of different traffic signal settings in some specified region of a large network. The initial experiment was on a section of Washington, D.C., which contained 80 signalized intersections. A macroscopic view of vehicle behavior was adopted. The simulation scan interval indicated was five seconds. Traffic lanes were divided into zones. The length of a zone was such that a car moving at free speed moved from one zone to the next in one time period.

Francis and Lott (11) have reported another model developed for simulating a general network. This program was applied to a series of nine traffic signals along a main traffic route in Central London. Although general in structure, the model was not capable of handling large networks such as the Washington, D.C. application.

During that same year, a third network model was reported by Rhee (32). Although few details of this model are available, it seems quite
similar to the TRANS model.

GPSS has found numerous applications in network simulation. Blum (4) proposed that intersection modules be used as generalized building blocks for the formation of any network configuration. This initial work was expanded (5) until now it is a general purpose network model capable of useful application to practical problems. In 1966, Schwartz (34) utilized GPSS II in programming a general purpose network model, which was applied to a segment of downtown Boston. Voskoglov and Wheeler (37) also made use of GPSS to study the effects of proposed parking facilities adjacent to the University of Missouri campus. Although this model represented a network application, the model developed was not a general-purpose model.

Sakai and Nagao (33) produced a model to simulate large networks. This program was field tested on a small area of Kyoto, Japan. A traffic display board was constructed for this experiment to provide a visual means of validating the model. The model divided the roadway into 50-meter blocks capable of containing a number of vehicles. The number of vehicles advanced was related to a transfer function and depended on relative density. A scan interval of four seconds was utilized.

The latest network model is reported by Lieberman, Worrall and Bruggeman (29) and is called UTCS-1. It is a microscopic simulation model designed to evaluate urban traffic control systems. The capabilities of the model include the ability to model vehicle-pedestrian conflicts, bus route and bus stop action, and intersection spill-back.
Freeway and Highway Simulation

Concurrent with the development of intersection and network models was the investigation of freeway traffic flow. Gerlough (12) reported the first application of simulation to freeway traffic. He studied a quarter-mile section of one-way, two-lane traffic. Each car had a preferred velocity, which was maintained when possible. Cars traveling below their desired speed were permitted to pass slower vehicles. As in earlier intersection work, a series of binary digits indicated positioning along the roadway.

The Midwest Research Institute (17) utilized simulation to study the interchange design problem. Actual data from four Chicago interchanges were used. Gap acceptance decisions were determined by distributions empirically derived. The freeway was divided into a series of blocks 17 feet long. During each second of simulation time, basic rules of advancement were checked to determine the new position of the vehicle.

Wohl (41) discussed the freeway merger problem as an illustrative example of simulation application. His basic model differs considerably from that of the Midwest Research Institute in that it was more concerned with time spacing than with actual position.

Howat (21) developed a model to investigate the driver-automobile interactions with respect to overtaking, following and passing on the open road. A decision table was used for modeling driver behavior. In 1966, Cassel (8) also studied overtaking and passing on the open road. Drivers were permitted to abort a passing attempt and accidents could
occur if the automobile could not respond quickly enough to new conditions.

Buhr, Besserole and Drew (7) reported the simulation of a section of freeway with multiple on and off ramps. The model was general in nature and could be applied to different situations. The overall structure of the program is similar to that of the Midwest Research Institute. However, of particular interest in the study was the use of a pre-loaded roadway.

Modeling Philosophies

Inherent in the simulation models developed to date and in the four models to be described in the next chapter are different philosophies for performing the actual simulation procedure. It is the purpose of this thesis to identify these philosophies and to establish their effects on model performance, speed and size. Before describing the models that were studied, different methods of delineation will be reviewed.

Physical Representation Versus Memorandum Representation

Gerlough (14) was one of the first researchers to attempt to classify different modeling approaches. He divided the then existing models into two general categories—physical representation and memorandum representation.

With the physical representation approach, one or more binary digits are assigned to represent static location blocks in a roadway. The data contained in these blocks can represent presence, position and possibly size of a vehicle on a specific segment of roadway.
computer memory are organized to represent the network being studied. Vehicles are advanced by manipulating the data in the static location blocks.

The memorandum method focuses attention on the individual vehicles. The characteristics and status of the vehicle form the basic unit of the data structure. Vehicles are advanced by changing the values of specific status variables such as velocity and position according to some prescribed logic.

**Time Scanning Methodologies**

For the purposes of this thesis, a more useful delineation can be obtained by expanding the work of Kell (26). Kell emphasized the importance of the time scanning methodology by first classifying the models as either fixed time scanning or next event scanning. Then he divided the fixed time scanning models into models which utilized a discrete roadway representation and those employing a continuous roadway representation.

With the fixed time scan methodology, the state of the system being simulated is examined at regular intervals of time. Vehicle position and motion characteristics are then updated and the process repeated at the next point in time. The actual position of the vehicle may have a discrete representation, causing the vehicle to move along the roadway by jumping from one point to another, or it may closely approximate a continuous representation.

The event scan methodology relates the actions of vehicles to specific points in time. Examples of these events are system entrance,
queue joining, intersection entrance and turn completion. Since each event considered in the scanning procedure has some known or determinable position associated with it, the sequence of events determines the sequence of vehicle positions. Position, therefore, does not need to be defined, and the event time of occurrence describes the necessary space/time relationships of vehicles.

The models described in this thesis can be classified according to Kell's scheme. Three fixed time scanning models are described which utilize different methods of representing the vehicle/roadway relationship. The final model is characteristic of the event scanning methodology.
CHAPTER III

DESCRIPTION OF THE MODELS

Common Features of the Models

A number of features were common to most, if not all, of the models developed in this research. A description of these features will precede detailed discussions of the individual models.

The General Form of the Network

Since it was desired to be able to simulate a variety of different situations, a general form of network representation was adopted. In all four models a traffic network is considered to be a set of links and nodes. A node represents a traffic intersection, a traffic source or a traffic sink. A link is a unidirectional road segment between two network nodes. Therefore, a one-way street between two intersections would be regarded as a single link while a pair of links would be necessary to represent a two-way connecting street. Figure 1 shows an illustrative network consisting of 12 nodes and 24 links.

Network nodes are classified as either external nodes or internal nodes. The external nodes act as traffic sources and sinks and may not represent actual network intersections. In Figure 1 nodes 1 through 8 are external nodes. The internal nodes represent intersections in the real network and are exemplified by nodes 9 through 12. For the purposes of this study, all internal nodes are assumed to be signalized with a fixed time signal controller. Signal parameters, such as cycle
Figure 1. Illustrative Traffic Network
length, green time, and offset, are required input data for each signal.

Each link is identified by stating the node number associated with it or by a unique index. For example, in Figure 1, link 24 goes from node 11, the tail or source node, to node 12, the head node. The index for the head node is also the signal index. The characteristics of each link must be supplied as input data. These include length, width (in lanes) and the turn probabilities associated with the head node.

To simplify matters somewhat and to provide consistency between the different models, a standard input format was adopted for all four models and is described in detail in Appendix A. Comments regarding assumptions and restrictions on network characteristics will be made throughout the remainder of this thesis.

**Pseudo-Random Number Generation**

The pseudo-random number generation method used is known as the multiplicative congruential method. The general procedure is described as follows. Let

\[ X_n = A \cdot X_{n-1} \pmod{m}, \]

where \( A \) is a constant multiplier,

\( m \) is the modulus, and

\( X_0 \) is the initial random number seed.

The sequence, \( \{X_n\} \), approximates a sequence of integers uniformly distributed between zero and the modulus. By dividing \( X_n \) by the modulus,
a sequence \( \{R_n\} \) of pseudo-random numbers is obtained.

**Vehicle Generation**

The time between arrivals of vehicles at an input node is assumed to be an exponentially distributed random variable. This distribution has been found to be fairly satisfactory at low volumes, but is usually inaccurate at moderate and high volumes. The significance of this assumption is examined in Chapter IV.

A separate random number sequence is used for each generation node. One additional random number sequence is used for all other Monte Carlo techniques. To insure that the volumes generated by each input node were within acceptable limits, the vehicle generation segment was isolated from the programs and five hours of generation were conducted at volumes of 200, 300, 400, 500, and 600 vehicles per hour. The number of arrivals per five-minute time period were recorded and tested against a Poisson distribution. Chapter IV summarizes the results for the selected random number seeds.

**Turning Behavior**

The destination of a vehicle on a given link is determined by a Monte Carlo technique. The probabilities of right turns, straight-through movements and left turns are specified as input for each link in the network. Three of the models, which treat vehicles individually, assign a desired movement to each vehicle as it enters the link. This desired movement is then used to select an appropriate lane on multi-lane links. The fourth model assigns the turning movement when the vehicle reaches the intersection.
**Speed Distribution**

The desired velocity of an individual vehicle is determined by a Monte Carlo technique for those models which treat vehicles individually. When the vehicle is created, it is assigned a desired velocity which remains fixed throughout its journey in the network.

By using a distribution of desired velocities, these models introduce a source of delay which is not found in models where a single desired velocity is assumed for all vehicles. Vehicles which are not affected by the intersection control device or by preceding turning vehicles may still be delayed by the need to slow behind a vehicle with a lower desired velocity. This source of delay depends not on the intersection control scheme but on the standard deviation of the desired velocity and the length of an individual link.

The exclusion of this distribution was thought to detract from the flexibility and realism of the models and it was therefore included where possible. To minimize the resulting differences in delay prediction, all models which included a speed distribution were forced to assume the same form. A slightly skewed representation of a normal distribution was used for all models that permitted a variable desired velocity.

**Left Turn Gap Acceptance**

The gap acceptance procedure utilized by all four models is a Monte Carlo technique for assessing gaps in the opposing flow. A random number is compared to the probability of accepting the existing gap. If the probability value is greater than the random number, the gap is
accepted. The distribution used is indicated by Figure 2 and is an approximation of distributions reported in the literature (36). No distinction is made between a gap and a lag.

Measures of Effectiveness

The models include as the principal part of their output certain measures of effectiveness that were thought to be important in evaluating system performance. The primary measures of effectiveness are:

- System Delay
- Queue Lengths
- Number of Stops

The definition and method of obtaining data on each of these measures will be deferred to the individual model presentations. The differences in interpretation is felt to be a major contributor to lack of consistency between the models (26).

The Continuous Model

The most detailed traffic simulation models have approximated a position of a vehicle by a continuous variable. The position of the vehicle is changed by using laws of rectilinear motion and the current state of the system. A continuous roadway representation introduces little rounding error due to positioning; however, rounding error due to time scanning still exists.

Lewis (28) and Gerlough and Wagner (16) utilized a continuous representation for modeling a single intersection while Buhr, et al. (7) chose the continuous approach to model freeway exit and entrance ramps. For these studies a detailed description of vehicle interaction was
Figure 2. Gap Acceptance Distribution
thought necessary to measure vehicular delay and to obtain realistic acceleration and deceleration procedures. The continuous model employed in this study will be referred to as the CONT model and draws heavily from the model developed by Lewis.

Roadway Representation

In the CONT model, each lane is a one-dimensional coordinate system. The link entrance is assumed to be the projection of the curb line at the tail of the link and is designated as the zero coordinate point. Link length is the distance, in feet, from this point to the projection of the near side curb line at the head node. The stop line is located 12 feet back from the extension of the curb line.

The coordinate system for each lane is extended into the intersection, since vehicles do not exit one link until they pass the entrance point of the next link. The coordinates of the end points are dependent on the intersection geometry and turning movement desired. In Figure 3 the end points of a one-lane and two-lane approach are illustrated. The complete lane stationing is included in Table 1.

Table 1. Intersection End Points

<table>
<thead>
<tr>
<th>Number of Lanes and Movement</th>
<th>Distance to End Point from Near Side Curb Lane (Feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Lane Approach</td>
<td></td>
</tr>
<tr>
<td>Left Turn</td>
<td>40</td>
</tr>
<tr>
<td>Straight Movement</td>
<td>30</td>
</tr>
<tr>
<td>Right Turn</td>
<td>12</td>
</tr>
<tr>
<td>Two-Lane Approach</td>
<td></td>
</tr>
<tr>
<td>Left Turn</td>
<td>85</td>
</tr>
<tr>
<td>Straight Movement</td>
<td>60</td>
</tr>
<tr>
<td>Right Turn</td>
<td>12</td>
</tr>
</tbody>
</table>
Figure 3. Link Endpoints for Two Types of Intersections

a. Two-by-Two Intersection

b. Four-by-Four Intersection
Because end points are a function of intersection geometry, the program must contain data on all possible intersection configurations to be included in the network. Since only two-lane by two-lane and four-lane by four-lane intersections were included in this study, the table is quite small. Expansion of the table would not be a major task if additional intersection geometries were included in the network.

**Vehicle Representation**

The vehicle in the CONT model is represented by a set of 12 computer words. These words contain data on the characteristics and status of the individual vehicle. Specific data are maintained on:

1. Desired Speed
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Estimated Link Exit Time
6. Position
7. Actual Speed
8. Index of the Vehicle in Front
9. Index of the Vehicle in Back
10. Stopping Tag
11. Acceleration Tag
12. Holding Movement Distance

As indicated earlier, the desired speed is a fixed characteristic of the vehicle and remains constant throughout the travel of the vehicle in the network. Items 2 through 5 are assigned values as the vehicle enters a new link. Items 6 and 7 are updated every time scan period based on the assumed motion of the vehicle during that time interval. A one second scan period was adopted. Items 8 and 9 provide a means of establishing the ordering of the vehicles on a given link. For each lane of each link a record identifying the first vehicle and the last vehicle is maintained. By starting with the first vehicle on a specific
link and examining the information contained in item 9, the second vehicle can be identified. This procedure can be repeated to determine the proper ordering of all vehicles on that link. Items 10 through 12 are associated with intersection movement.

**Vehicle Movement**

The heart of any continuous model is the car-following vehicle behavior relationships. Car-following theories attempt to relate the reaction of a following vehicle to the motion of a leading vehicle.

One form of the spacing-speed relationship assumes that spacing is a linear function of speed. Lewis used this relationship to structure his model behavior. A second form of car-following behavior proposes that the response of a following driver is the product of a sensitivity factor and a stimulus. Chandler, Herman and Montrol (9) suggest that the stimulus be the relative velocity of the two vehicles and that a response of acceleration or deceleration be delayed some constant reaction time. Gerlough and Wagner (16) used this function with the addition of an inverse relationship to vehicle spacing. The CONT model is a generalization of the linear space/speed relations used by Lewis. A detailed development of the equations to follow can be found in Reference 28.

**Acceleration Restriction.** It is assumed that all vehicles attempt to travel at some desired speed, which is a characteristic of each vehicle. A uniform rate of speed change is assumed under free flow conditions. A vehicle, otherwise unrestricted, will accelerate to its desired velocity at an acceleration rate $\ddot{A}$ of $4$ ft/sec$^2$. Let $ZA$ be the distance
that a vehicle travels in one second based on this restriction. Then

\[ ZA = \min\{V_{t-1} + 0.5A, 0.5(V_{t-1} + V_{\text{max}})\}, \]

where \( V_{t-1} \) is the velocity of the vehicle one second ago,
\( V_{\text{max}} \) is the maximum desired velocity for the vehicle and,
\( A \) is the desired constant acceleration rate.

Once at the desired velocity, the vehicle will maintain that velocity until some other restriction is encountered.

**Car-Following Restriction.** When the spacing between two vehicles is critical, the speed of the second vehicle is adjusted so that at the end of a time scan the new position is no closer than desired. The desired spacing assumes that vehicles moving at the same speed will maintain a spacing linearly proportional to their velocity and a fast overtaking vehicle will keep sufficient distance to permit a safe stop.

Using these assumptions, Lewis has derived the maximum distance a vehicle can advance in one second based on the car-following restrictions. Let \( ZS \) be the maximum distance a vehicle can move in one second.

Then when \( V_{t-1} > V_t' \)

\[ ZS = 0.5(V_{t-1} + V_t') - 0.75D + \left[ 0.5625D^2 - 0.25D^2 V_{t-1} + 0.75V_{t-1} + 0.5(X_t' - X_{t-1} - P) \right]^{1/2}, \]

and when \( V_{t-1} \leq V_t' \)

\[ ZS = \left[ X_t' - X_{t-1} - P + V_{t-1} \right]^{1/3}, \]
where

\[ Z_{S} \] is the maximum distance a vehicle can move in one second,
\[ V_{t-1} \] is the velocity of the following vehicle one second ago,
\[ V_{t}' \] is the present velocity of the lead vehicle,
\[ X_{t-1} \] is the position of the following vehicle one second ago,
\[ X_{t}' \] is the present position of the lead vehicle,
\[ \bar{D} \] is the desired deceleration rate, and
\[ P \] is the assumed minimum spacing obtained at zero velocity.

**Stopping Restriction.** As a vehicle approaches an intersection, the traffic control device may restrict its movement. When a traffic signal turns from green to amber, the associated input links are "tagged." The first vehicle in each link which can stop within reasonable deceleration limits is then tagged and it begins decelerating to a complete stop. Preceding vehicles in that lane are allowed to continue through the intersection while those vehicles following the tagged vehicle will stop as a result of the car-following restriction. Only one vehicle per lane is tagged. When a traffic signal turns to green, all associated input tags are removed.

Let \( Z_{D} \) be the maximum distance a vehicle operating under the stopping restriction may advance during a one-second period. Then

\[
Z_{D} = 0.5V_{t-1} - 0.25\bar{D} + \left[ 0.0625\bar{D}^2 - 0.25\bar{D}V_{t-1} + 0.5DX_{\text{gap}} \right]^{\frac{1}{2}},
\]

where

\[ Z_{D} \] is the maximum distance under the stopping restriction,
\( V_{t-1} \) is the velocity of the vehicle one second ago, 
\( \bar{D} \) is the desired deceleration rate, and

\( X_{\text{gap}} \) is the distance from the vehicle to the stopping point.

If the intersection is blocked by a vehicle from a side street or by a left turning vehicle from the opposing link, a stopping restriction is applied to the first vehicle of each lane. The stopping point for these blockages is assumed to be one foot beyond the extension of the curb line.

**Turning Restriction.** During a turning maneuver it is assumed that a vehicle will decelerate to some desired speed then accelerate throughout the remainder of the movement. The point of inflection in the acceleration pattern is called the "turn point" and the location of the turn point is assumed to be a function of both intersection geometry and desired movement as indicated by Table 2.

<table>
<thead>
<tr>
<th>Number of Lanes and Movement</th>
<th>Distance from Near-Side Curb Line to Turn Point (Feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>One-Lane Approach</strong></td>
<td></td>
</tr>
<tr>
<td>Left Turn</td>
<td>7</td>
</tr>
<tr>
<td>Right Turn</td>
<td>1</td>
</tr>
<tr>
<td><strong>Two-Lane Approach</strong></td>
<td></td>
</tr>
<tr>
<td>Left Turn</td>
<td>13</td>
</tr>
<tr>
<td>Right Turn</td>
<td>1</td>
</tr>
</tbody>
</table>

Let \( ZT \) be the maximum distance which a vehicle can travel in one second under the turning restriction. Then
\[ ZT = 0.5V_{t-1} - 0.25\bar{D} + \left[0.0625\bar{D}^2 + 0.25(V_{TP}^2 - \bar{D}V_{t-1}) + 0.5\bar{X}_{gap}\right]^{\frac{1}{2}}, \]

where \( V_{t-1} \) is the velocity of the vehicle one second ago,
\( \bar{D} \) is the desired deceleration rate,
\( V_{TP} \) is the maximum permissible turn point velocity, and
\( \bar{X}_{gap} \) is the distance between the vehicle and the turn point.

This equation is only applicable when the vehicle does not proceed beyond the turn point. When this occurs a new value for \( ZT \) must be computed which will consider the assumed deceleration and acceleration.

Left turning vehicles will not proceed past the turn point unless an acceptable gap is found in the opposing stream of traffic. Each lane of the opposing link is searched to find the position and velocity of the first vehicle. The time gaps are then calculated for each lane and the minimum gap used to make a decision. If the minimum gap is accepted the vehicle proceeds through the turn point; if rejected the vehicle immediately stops at the turn point to wait for an acceptable gap. The gap acceptance table is then checked every time scan until an acceptable gap is found. Left turning vehicles which have passed the turn point accelerate at 7, 6 and 5 ft/sec\(^2\) for the first three seconds after an acceptable gap becomes available.

**Measures of Effectiveness**

The CONT model provides output on system delay, maximum queue lengths and number of stops. All statistics are printed by individual link, by intersection and by total system performance.

Delay is defined as the difference between actual travel time and
undelayed travel time. Since the model assigns each vehicle a desired velocity from a specified probability distribution, undelayed travel time must be calculated for each vehicle. Furthermore, a vehicle which has just entered a link may not, and most likely will not, start at coordinate zero. The actual starting location will depend on the distance the vehicle was from the end point of the previous link at the beginning of the scan time and how far it advanced during that scan cycle.

To calculate undelayed travel time, a single vehicle was permitted to traverse a link and the travel time was recorded. Link length, desired velocity and desired turning movement were varied in order to obtain travel time as a function of these factors. As each vehicle enters a new link, the initial position, link length, desired speed and desired turning movement are considered in calculating the undelayed exit time. Upon exiting the link, this undelayed exit time is subtracted from the actual exit time to obtain delay.

If a vehicle slows to a speed less than 4 ft/sec, it is assumed to stop and the stop is recorded. Lewis (28) provided a means of measuring stop delay but this was not incorporated in the CONT model.

The estimation of queue length was difficult in this model. The accordion effect often witnessed in heavy urban traffic can exist in this model. Hence, a vehicle far removed from the intersection may be stopped, while vehicles in front of this stopped vehicle are accelerating. Since it was desired to estimate the maximum queue lengths obtained, the length of a queue was recorded each time the signal at the head of the length turned green. A routine was written to scan each
lane and find the last stopped vehicle. All vehicles in front of this vehicle were considered to be in the queue, even if they were moving greater than 4 ft/sec.

**General Model Flow**

The general flow of the model is indicated by Figure 4. A complete listing of the program is given in Appendix B. The initialization phase, which is common to all four models, consists of the four subroutines:

- **EXTERN** - to read data cards on external nodes.
- **INTERN** - to read data cards on internal nodes and signals.
- **INFLKS** - to read data cards on all network links.
- **SETUP** - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation consists of four primary subroutines, as indicated in Figure 4, and 12 secondary subprograms.

The **SIGCHK** subroutine checks the signal at each internal node to see if a change of state is desired. A change from green to amber will cause the appropriate input links to be tagged as described earlier.

The **VEGEN** subroutine checks each external node to see if it is time to create a new vehicle. If this is desired, a new vehicle is created and the next vehicle to enter is scheduled. The new vehicle is placed in the "holding area" of the appropriate link. This procedure will be explained shortly.

The **LKMOVE** subroutine is concerned with the movement of vehicles on the links and through the intersections. Each node in the network is checked individually. Input links for the node are examined one at a
EXTERN
• Reads data on external nodes.

INTERN
• Reads data on internal nodes and signals.

INFLKS
• Reads data on all links.

SETUP
• Establishes network relationships such as link destinations, etc.

Increments CLOCK and sees whether run is over

SIGCHK
• Checks each signal for change and gathers queue data if turned green.

VEGEN
• Checks external nodes for input and creates vehicle when ready.

LKMOVE
• Starts with first link and processes all links for vehicle movement.
• Applies all appropriate restrictions and moves vehicle smallest distance.
• Terminates or advances vehicle to new link as needed.

HOLDOU
• Takes vehicle out of holding area and places it on link.

Yes

STAT
• Prints statistics by link, intersection and total system.

STOP

Figure 4. General Flow of CONT Model
time. Vehicles are processed starting with the first vehicle in the left-most lane and proceeding sequentially down the lane from the head to the tail.

As each vehicle is examined, the maximum possible distance it could advance is calculated for each of the four restrictions. The minimum of these distances is assumed to be the desired advance. If the desired distance will take the vehicle past the turn point (past the near side curb line for straight movement), the destination link is checked for available space before proceeding into the maneuver.

If the new position of the vehicle is beyond the end point of the link, it is removed from the link and placed in the "holding area" of the destination link. After all links have been processed, each holding area is checked for vehicles. These vehicles are then removed from the holding areas and allowed to start down their new links. This discontinuity of processing is necessary to avoid processing a vehicle twice in the same scan interval--once on the old link and then on the new link. Neither single intersection models nor arterial models present this problem, which results from the application of a general model to a closed loop network.

The purpose of the HOLDOU subroutine is to remove vehicles from the holding areas after all link movement is completed. Additionally, each vehicle is assigned a turning maneuver and an expected exit time.

The Unit Block Model

Many of the early efforts in traffic simulation relied on a discrete representation of vehicular positioning. Goode, Pollmar and
Wright (18) considered the roadway as a string of distinct points where vehicles were permitted to jump from point to point. Wong (42) proposed a slightly more general model by dividing the roadway into a series of blocks. A roadway segment one car length long and one lane wide was called a unit block. Each unit block could contain one car or it could be empty. Cars were confined to speeds of integral numbers of unit blocks per unit time.

Stark (35) developed a unit block model where blocks were 12 feet in length. The positioning of the vehicle within the block was to the nearest 1/100th of a block and vehicles could advance in the quarter second scan without changing blocks. The precision of Stark's model closely approximates that of a continuous model. The model developed for this study is patterned more closely to the Wong model than that of Stark or Goode et al. It will be referred to as the UB model.

**Roadway Representation**

The UB model divides the roadway into a series of 18-foot blocks, each one lane wide. The unit blocks are organized into an array in the computer memory. The presence or absence of a vehicle in a unit block can be determined by examining the value contained in the appropriate memory location. A zero implies that the block is empty while a positive number indicates the index of the vehicle occupying that specific roadway position.

The unit block notation provides a means for identifying and ordering the vehicles and specifying their location. Figure 5 illustrates two one-lane links connecting two intersections. By starting at the
Figure 5. Unit Block Representation of a Two-Lane Road Segment
head of the link and working back to the tail, all vehicles on the link can be identified and ordered.

The representation of an intersection using a unit block approach requires additional logic. Stark (35) indicates that it was necessary to superimpose unit blocks from crossing streets. The UB model partitions the intersection into distinct intersection squares and requires a complex algorithm to associate unique computer memory locations with each square. This approach was taken to insure that two vehicles on different links were never, in reality, on the same segment of an intersection. Figure 6 identifies the numbering of intersection squares for a two-lane by two-lane and a four-lane by four-lane intersection. The Roman numerals identify the approach orientation for the given block numbering scheme.

If additional intersection geometries were necessary for a specific application each unique geometry would require an explicit definition of the intersection partitioning.

Vehicle Representation

The vehicle in the UB model is represented by a set of nine computer words. The specific records maintained on each vehicle are:

1. Desired Speed (In Blocks Per Second)
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Estimated Link Exit Time
6. Actual Speed
7. Move Indicator
8. State Indicator
9. Holding Movement Distance

Since the desired speed is integral in block length, a limited number of speeds are possible. Assuming a block length of 18 feet, the
Figure 6. Intersection Diagrams for the UB Model

a. Two-Lane by Two-Lane Intersection

b. Four-Lane by Four-Lane Intersection
speed distribution assumed is indicated in Table 3. This same distribution was specified in those models which allowed variable desired speeds. It somewhat represents a normal distribution suggested by Gerlough and Wagner (16).

Table 3. Distribution of Desired Speeds

<table>
<thead>
<tr>
<th>Desired Speed</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks/Sec</td>
<td>Feet/Sec</td>
</tr>
<tr>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
</tr>
</tbody>
</table>

Items 2 through 6 and item 9 have identical counterparts in the CONT model described earlier. The move indicator is set to one if the vehicle has been processed during the current scan; otherwise it is set equal to zero. The state indicator is a four state switch indicating whether the vehicle is accelerating, decelerating, traveling at a constant velocity or standing still.

Vehicle Movement

The complexity of the car-following equations found in the CONT model are avoided in the UB model; however the restrictions are very similar.

Intra-Link Movement. An otherwise unrestricted vehicle will accelerate to its desired velocity at the rate of 1 block/sec². This rate is considerably faster than the acceleration limit assumed in the CONT model. Therefore a restriction is imposed which limits
acceleration to every second scan cycle. This restriction reduces the average acceleration rate to $9 \text{ ft/sec}^2$.

After the new position based on the current velocity of the vehicle and the acceleration restriction has been calculated, the headway between this position and the next vehicle is examined. A table of minimum headways is referenced to insure that an adequate separation is being maintained. These minimum headways are a function of the relative velocities of the two vehicles as shown in Table 4. They represent a compromise between the headways found in the CONT model and the minimum safe following distances possible with a deceleration rate of one block per second.

Table 4. Minimum Acceptable Headway

<table>
<thead>
<tr>
<th>Following Vehicle's Speed (Blocks/Sec)</th>
<th>Lead Vehicle's Speed (Blocks/Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4</td>
</tr>
<tr>
<td>1</td>
<td>0 1 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>1 2 2 0 0</td>
</tr>
<tr>
<td>3</td>
<td>3 4 4 2 0</td>
</tr>
<tr>
<td>4</td>
<td>6 6 4 4 3</td>
</tr>
</tbody>
</table>

Vehicles desiring to turn must also check their new position with respect to the intersection. All turning vehicles are required to slow to one block per second velocity in the intersection area. A table look-up procedure establishes the maximum permissible turn velocities for various distances from the intersection.
All vehicles approaching an intersection with an amber or red light are required to stop. A table look-up procedure is employed to establish the maximum velocity possible based on the vehicle's current position.

If the advancement of the vehicle will carry it into the intersection, additional checking is required. A vehicle will not enter an intersection unless there is space available in the destination link. Furthermore, vehicles in the left-most lane will not enter the intersection if there is a left-turning vehicle in front. This restriction is necessary to insure that intersections will not become hopelessly jammed by opposing left-turn vehicles. Headway restrictions apply between vehicles in the intersection and vehicles that are attempting to enter the intersection.

**Inter-Link Movement.** Movement restrictions within an intersection depend on the turning maneuver desired and the intersection geometry. All turning vehicles are restricted to a 1 block/sec velocity for the first half of the maneuver. Once the turn is executed, a normal 1 block/sec^2 acceleration is assumed.

Figure 7 illustrates the paths followed by vehicles making different maneuvers. The sequence of intersection squares encountered is a function of direction of approach as well as intersection geometry and desired turning maneuver.

A left-turning vehicle advances two or three squares into the intersection before observing the gap in the opposing flow. Because of the discrete nature of velocities and locations the model gaps are only an
Figure 7. Intersection Maneuvers for the UB Model
approximation of the actual gaps. However, the error due to this approxima-
tion is not greater than .5 seconds, the intrinsic error caused by the nature of the model. If a left-turning vehicle rejects the gap and stops, the inter-link routine maintains an empty block behind the waiting vehicle to permit opposing left-turning vehicles to clear the intersection.

*Measures of Effectiveness*

The UB model provides output on system delay, maximum queue length and number of stops. All statistics are printed by individual link, by intersection and by total system performance.

Like the CONT model, the UB model calculates delay by subtracting the undelayed travel time from the actual travel time. Undelayed travel times were obtained by taking vehicles with different desired speeds through all combinations of turning maneuvers and intersection geometries.

A stop is defined as the act of going from a non-zero velocity to a velocity of zero. All stops are recorded by lane for each link in the network. Stop time delay is easily obtained from this type of model; however, it was not included in this study since it is not easily available with the other models included.

The queue length is evaluated for a link each time the light turns green. The largest queue found is assumed to be the maximum queue. To evaluate the length of the queue, each block, starting with the last block, is examined until a stopped vehicle is found. All vehicles in front of this vehicle are considered to be in the queue.
General Flow Model

The general flow of the model is illustrated in Figure 8. A complete listing of the FORTRAN program is given in Appendix C. The initialization phase consists of the four subroutines:

EXTERN - to read data cards on external nodes.
INTERN - to read data cards on internal nodes and signals.
INPLKS - to read data cards on all network links.
SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation consists of five primary subroutines and 13 secondary subprograms. The basic structure of all three fixed time scanning models is very similar.

The SIGCHK subroutine checks the signals at each node to see if a change of state is desired. A change from red to green will cause the queue lengths to be examined on the appropriate inputs.

The VEGEN subroutine checks each external node to see if it is time to create a new vehicle. If this is desired a new vehicle is created and placed into the holding area of the appropriate link.

A separate subroutine was written to process inter-link movement for each intersection geometry considered in the network. The INTERL subroutine is a control routine which takes intersections one at a time and passes them to the appropriate processing routines. The MOVETT subroutine processes vehicles for a two-lane by two-lane intersection while the MOVEFF subroutine processes vehicles for a four-lane by four-lane intersection.

The intersection is first checked for vehicles from the non-green
Figure 8. General Flow of UB Model
approaches. Then the green approach vehicles are considered for advancement. Within each approach scan, vehicles completing left or right turns are processed first. Then the lane extensions are scanned from destination to the start of the link.

If it were desired to model different intersections such as a two-lane by four-lane or an intersection with a one-way approach, a separate subroutine would be necessary for each unique configuration. A completely general network simulator using the concepts presented in the UB model would require a substantial amount of additional programming to apply to these different conditions.

The INTRAL subroutine processes vehicles traveling between two nodes. The scanning process begins with the first block at the head of the link. Each block on the link is inspected for the presence of a vehicle. If a vehicle is found, the movement restrictions are applied and the vehicle is advanced the desired number of blocks. All blocks are searched until the last block on the link is processed.

The HOLDOU subroutine removes vehicles in the holding areas and places them on the appropriate links. To avoid processing the same vehicle twice in the same scan, the HOLDOU subroutine is not executed until after all intra-link movements have been considered.

The Zone Model

The two models presented thus far have been microscopic in nature in that vehicles are individually maneuvered through the network with somewhat complex decision processes being simulated every time scan. In order to model a larger, more complex system of traffic, a macroscopic
approach has been adopted by a number of researchers.

Gerlough et al. (13,22) developed the TRANS model to evaluate the effect of traffic signal settings on traffic flow in a region of a city. Initially it was applied to a section of the District of Columbia containing over 300 links and 80 intersections. At about the same time Rhee (32) developed a program (Urban Traffic Control Simulator) which utilized a representation very similar to the approach taken by TRANS. Sakai and Nagao (33) adopted a network representation similar to TRANS, but the movement of vehicles was somewhat different and will be described below. The zone model developed for this research is patterned after the TRANS model and will be referred to simply as the ZONE model.

Roadway Representation

In the ZONE model each lane is divided into zones as indicated by Figure 9.

Figure 9. Two Two-Lane Links Divided into Zones
The length of a zone is such that a car moving at "free-flow" speeds from one zone to the next is the time interval between scans. In the initial version of TRANS a five-second scanning cycle was adopted to take full advantage of the macroscopic nature of the program. The ZONE model assumes the two-second scanning period which was reflected in more recent applications of TRANS (39).

A two-second scanning cycle and a free-flow speed of 53 ft/sec (the mean of the distribution used by the other models) implies a 106-foot zone length. Finally, assuming an average vehicle length of 20 feet, the capacity of each zone is five vehicles. To test the effects of a two-second scan cycle, a specific situation was tested under varying scan cycles, with results reported in Chapter IV.

The physical intersection has no counterpart in the model. Vehicles are transferred from the first (exit) zone of the source link to the destination link. This movement makes intersection geometry immaterial and, as a result, link geometry considerations (number of lanes, turn lanes, etc.) present the only restriction to network representation.

The ZONE model utilizes the computer words associated with individual network zones to indicate the number of vehicles on specific segments of roadway, whereas the UB model utilizes the block to indicate the index of the vehicle concerned.

Vehicle Representation

The ZONE model is unique among the four models tested in that vehicles are treated in clusters rather than individually. Only in the critical area of the intersection does the ZONE model resort to the
individual handling of vehicles. As a result of this no computer memory
is allocated to storing data on vehicles.

Traffic Movement

The movement of traffic in the ZONE model is divided into inter-
link movements and intra-link movements. Inter-link movements are char-
acterized by a vehicle-by-vehicle consideration while intra-link move-
ments manipulate platoons of vehicles.

Inter-Link Movements. The inter-link movements transfer vehicles
from the first (exit) zone of the source link to the holding area of the
destination link. The ZONE model has no representation of vehicles
within an intersection. Vehicles are drawn from the exit zone one at a
time. A Monte Carlo technique is applied to determine the destination
link and the move is attempted.

An aborted exit will cause the model to stop processing the lane
and proceed to the next lane of that link or a new link, whichever is
appropriate. If a left turn fails to find an acceptable gap in the
opposing flow, a flag is set and the model will not process any inter-
link movements from that lane until the first vehicle completes a left-
turn maneuver.

The number of vehicles which may be transferred from an exit zone
depends on several factors. If the link is in the queue state, then a
discharge rate restricts the number of vehicles which can be transferred.
If the destination refused entry or if an acceptable gap is not found for
a left turn, the transferring is terminated. Otherwise, all vehicles in
the exit zone are assumed to move at the free-flow speed, 53 ft/sec, to
their desired destinations.

All vehicles in the exit zone are stopped by a red light. If the light is amber and if the turn flag indicates a vehicle is waiting for a left turn, that vehicle is permitted to traverse the intersection. All other vehicles are stopped by the amber light.

It should be pointed out that the method of roadway representation used in the ZONE model makes all gaps integral multiples of scanning time. For larger scanning cycle times, this assumption is much coarser than the acceptance graph indicates. For example, with a scan time of 5 seconds, only gaps of size 0, 5, 10, 15 and 20+ seconds are possible.

Intra-Link Movement. Once the first zone of a link has been processed, the remaining zones are processed by a very simple algorithm. The model advances as many vehicles from the \((j+1)th\) zone as can be accommodated in the \(jth\) zone. Any vehicles not advancing are considered to have been delayed.

Sakai and Nagao (33) proposed a more complex scheme for advancing vehicles from zone to zone. A transfer formula was specified which related the number of vehicles advanced to the relative speed of the two zones concerned. The zone speed was estimated based on block density. Figure 10 indicates the propagation of a traffic stream under this formula. Although this approach seems desirable, the approach employed in the TRANS model was incorporated in the ZONE model. This approach allows queue discharge restrictions to spread out a platoon and are computationally more efficient.
Figure 10. Platoon Dispersion Utilizing Transfer Formula
Measures of Effectiveness

The ZONE model provides output on average delay and maximum queue lengths. All statistics are printed by individual link, by intersection and by total system performance. Statistics on turning maneuvers are not recorded.

As network zones are processed, all vehicles are either advanced or delayed. If delay occurs in a zone, the number of vehicles delayed is recorded. The number of vehicles processed on a link is divided into the cumulative delay to obtain average delay. Field studies (38) indicate that this measure is adequate for light-to-medium traffic; however, average delay estimates are lower than expected in heavy traffic.

The number of vehicles in the front zone of a given link is examined whenever the traffic signal turns green. All vehicles in this zone are considered to be in the queue. If the exit zone is full, the next zone is examined. If this is full all the vehicles are considered to be in the queue and the next zone is examined. The first non-full zone marks the end of the queue. If this zone is below half full, the vehicles are considered to be moving and not in the queue; otherwise, the vehicles are the last members of the queue.

General Model Flow

The general flow of the ZONE model is illustrated in Figure 11. A complete listing of the program is given in Appendix D. The initialization phase consists of four subroutines.

EXTERN - to read data cards on external nodes.

INTERN - to read data cards on internal nodes and signals.
EXTERN
• Reads data on external nodes.

INTERN
• Reads data on internal nodes and signals.

INPLKS
• Reads data on all network links.

SETUP
• Establishes network relationships such as link destinations, etc.

INCREMENTS CLOCK and sees whether run is over.

SIGCHK
• Checks each signal for change and gathers queue data if turned green.

VEGEN
• Checks external nodes for input and creates vehicles when ready.

INTERL
• Takes platoon in exit zones and advances vehicles individually to new link.

INTRAL
• Performs zone to zone movement on each link.

HOLDOU
• Takes vehicles out of holding area and places them on link.

STAT
• Prints statistics by link, intersection and total system.

STOP

Figure 11. General Flow of ZONE Model
INPLKS - to read data cards on all network links.

SETUP - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation program consists of five primary subroutines and five auxiliary subprograms.

The SIGCHK subroutine checks the signal at each intersection node to see if a change is desired. A change to green will cause the queue length to be evaluated on the appropriate input links.

The VEGEN subroutine checks each node to see if it is time to create a new vehicle. If this is desired, a new vehicle is created and the next vehicle to enter is scheduled. Because of the larger scanning times, a node may often input more than one vehicle at the same time. New vehicles are placed in the holding area of the appropriate link.

The INTERL subroutine processes vehicles in the first (exit) zone of each lane of all links while the INTRAL subroutine processes all other zones. The simplicity of these routines is such that only three supporting subprograms are necessary.

The HOLDOUT subroutine removes vehicles from the link holding areas and places these vehicles on their respective links. This is done by increasing the contents of the last zone by the number of vehicles in the holding area.

The Next-Event Model

The second major departure from the more classical concepts of the continuous and unit block models is a group of models known as next-event simulation models. By concentrating on the event sequence seen by a
vehicle instead of the position sequence, researchers hoped to improve the speed of the simulation program without sacrificing the accuracy of a model in predicting the desired measure of effectiveness.

Kell (23), Thomasson (36) and Wright (43) used event-oriented models to study various types of intersections. Wohl (41) suggested this approach to study the freeway merging problem, while Francis and Lott (11) used a similar technique in studying a traffic network.

A second set of users of the next-event concept is the group of researchers who have programmed their models in simulation languages such as GPSS. Schwartz (34) developed a GPSS model to simulate a general traffic network, then applied the program to a segment of Boston. Barnes (1) developed a set of GPSS programs to simulate different types of intersections. It is from these two works that a majority of the logic has been taken to develop the last program presented in this thesis. This program will be referred to as the NEXT model.

Roadway Representation and Traffic Movement

In an event-oriented model, vehicle actions are related to specific points in time rather than precise locations on the roadway. Since each event is associated with a different position along the roadway, position does not have to be explicitly defined. At the time of an occurrence of an event, the position of the vehicle is known. Between the occurrences of two events, the vehicle is in a transition state.

Between two intersections the actual roadway representation is a sequence of three events. Figure 12 illustrates these points for a one-lane link with the associated intersections.
The first point on the roadway marks the entrance of the vehicle to that link. The second point of interest is the queue entrance time. If no line exists, the vehicle instantaneously represents a queue of length one. When the vehicle reaches the front of the queue, a third event occurs in that it now becomes ready to enter the intersection.

The intersection is divided into squares and the vehicle is advanced from one square to the next. Point 4 marks the two possible exit points of the S1 square. The desired turning movement will determine which exit point is encountered. Point 5 marks the exit points for the S2 square and point 6 is the exit point for S3.

Table 5 specifies the different states possible for a vehicle. Some of the states have definite time durations associated with them, while others are delay states and are system dependent. For example, the time duration for crossing an intersection square depends on the desired turning movement, the type of intersection and an acceleration factor; while travel time down a link is a function only of the desired velocity.
and the link length. Time spent in a queue state is a natural result of the simulation execution.

Table 5. Possible States of Vehicle in NEXT Model

<table>
<thead>
<tr>
<th>State</th>
<th>Event</th>
<th>Description</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1-2</td>
<td>Free-flow in link</td>
<td>Formula</td>
</tr>
<tr>
<td>1</td>
<td>2-3</td>
<td>In queue, not first vehicle</td>
<td>Delay-system dependent</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Blocked at head of queue by light</td>
<td>Delayed-system</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Blocked by queue discharge rate or congested intersection</td>
<td>Delay</td>
</tr>
<tr>
<td>4a</td>
<td>4</td>
<td>Free-flow through S1</td>
<td>Formula</td>
</tr>
<tr>
<td>4b</td>
<td>4</td>
<td>Blocked by congestion at exit point</td>
<td>Delayed</td>
</tr>
<tr>
<td>5a</td>
<td>5</td>
<td>Free-flow through S2</td>
<td>Formula</td>
</tr>
<tr>
<td>5b</td>
<td>5</td>
<td>Blocked by congestion at exit point</td>
<td>Delayed</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Free-flow through S3</td>
<td>Formula</td>
</tr>
</tbody>
</table>

Transition from one state to another is based on the condition of the system and the associated rules, such as maximum queue discharge rate, intersection capacity and gap availability for left-turning vehicles.

Vehicle Representation

The vehicle is represented by a set of eight computer words.

These words contain:

1. Desired Speed
2. Present Link
3. Present Lane
4. Desired Turning Movement
5. Time Vehicle Entered Link
7. Time Vehicle is Scheduled to Change States
8. Index of the Next Vehicle on the Schedule Chain.

Items 1 through 4 are common to all of the models which treat vehicles independently. At the time of link entrance, the value of the simulation clock is stored for the entering vehicle and this time is used to calculate delay when the vehicle exits the link. Item 6 indicates the state of the vehicle while item 7 specifies the scheduled time of the next state change. Item 8 will be explained in the next section.

Vehicle Movement and the Event Chain

An ordered list or "chain" of all vehicles in the network is maintained. The ordering is by scheduled event time, item 7 of the vehicle record. Vehicles with the smallest scheduled event times are at the top of the list. Delayed vehicles maintain the event times they had when they entered the delay state. Thus they are above vehicles which are scheduled to change states at some point in the future.

The next vehicle to be processed is the first vehicle on the chain with an event time greater than or equal to the current simulation clock time.

The first vehicle on the chain is identified by a pointer. This record of this vehicle indicates the next vehicle on the chain by the last data item. By going from vehicle to vehicle the chain can be constructed. The process is identical to the technique used in the CONT model to establish the ordering of the vehicles on a specific link.

Vehicles are "moved" through the network by changing their records. When the simulation clock equals the event time, the model
attempts to change the state of the vehicle. The conditions which are necessary depend on the old state of the vehicle.

**Measures of Effectiveness**

The NEXT model provides output on average delay, number of stops and the queue lengths. All statistics are summarized by link, by intersection and by total system performance.

Delay is defined as the difference between actual travel time and undelayed travel time. Since all state durations other than delay are calculated from deterministic functions, the undelayed travel time is easily evaluated when the desired speed, link length and intersection geometry are known. If probabilistic functions had been used, the resulting delay estimates might not be meaningful. For example, if expected travel time were used to calculate delay and a vehicle drew a fast acceleration rate, the resulting delay might have a negative value. Additionally, the field data used to estimate the necessary parameters would have to be devoid of any delay, if the simulation model is to predict this delay through system performance.

Each time the vehicle enters a delay state, a stop is recorded. The maximum number of stops a vehicle can contribute to the cumulative total stops for a given link is a function of the turning movement desired. A left-turning vehicle would stop as many as three times while a right-turning vehicle could encounter one stop at most.

The NEXT model is the only model which continuously maintains statistics on the status of a queue. Since queue entrance time is not scheduled until the full length of the link has been traversed, this
statistic should be somewhat lower than that measured by the other models. A more complex algorithm might be used to calculate travel time to the end of the queue if this statistic were used as a primary measure of effectiveness. Such an algorithm would consider queue length, vehicle length, discharge rate and current state of light and time of next light change.

**General Model Flow**

The general flow of the NEXT model is illustrated in Figure 13. A complete listing of the program is given in Appendix E. The initialization phase consists of four subroutines:

- **EXTERN** - to read data cards on external nodes.
- **INTERN** - to read data cards on internal nodes and signals.
- **INPLKS** - to read data cards on all network links.
- **SETUP** - to manipulate data into desired form and schedule the first arrival for each input node.

The main body of the simulation program consists of three primary subroutines and nine auxiliary subprograms. As indicated in Figure 13, the organization and processing of these subroutines differs drastically from that of the other three models.

The main program examines three subroutine clocks to see when the next state change of some system component is going to occur. These clocks are:

- **SIGTIM** - the time of the next signal change.
- **VEGTIM** - the creation time of the next new vehicle.
- **MOVTIM** - the next vehicle event time.
EXTERN
• Reads data on external nodes.

INTERN
• Reads data on internal nodes and signals.

INPLKS
• Reads data on all network links.

SETUP
• Establishes network relationships such as link destination.

Sets CLOCK equal to min{VEGTIM, SIGTIM, MOVTIM}

Checks whether run is over.

Branches based on CLOCK.

CLOCK = SIGTIM
SIGCHK
• Updates first signal on signal chain and checks next signal for time of chain.

CLOCK = VEGTIM
VEGEN
• Creates vehicle for external node and checks next node for next arrival

CLOCK = MOVTIM
MOVVEH
• Processes all vehicles on vehicle chain which have a state change time less than or equal to value of CLOCK.

STAT
• Prints statistics by link, intersection and total system.

STOP

Figure 13. General Flow of NEXT Model
The simulation clock is advanced to the minimum of these three times. Control is then passed to the appropriate subroutine.

The SIGCHK subroutine changes the state of traffic signals when it is desired. Like the vehicles, signals are maintained on a chain in an event time ordering. This feature eliminates the necessity of checking each signal to see if a change is necessary.

The VEGEN subroutine creates vehicles and places them on the desired link. Because of the method of processing vehicles, no holding area is required in this model.

The MOVVEH subroutine manipulates vehicles which are scheduled to be processed at the current clock time or are in a delay state. If a congested link has prohibited a newly created vehicle from entering the system, the VEGEN routine is called to see if the congestion has dissipated.
CHAPTER IV

COMPARISON AND EVALUATION OF MODEL PERFORMANCE

Once the four models were programmed and debugged, an extensive testing program was undertaken to compare the behavior of the models under different situations.

Model Verification and Validation

Before the actual comparison of the models was considered, a number of computer runs were devoted to establishing the credibility of the programs. This operation was divided into two phases:

1. The verification stage, which was concerned with comparing the operations of the program against the desired logic of the model.

2. The validation stage, which was concerned with comparing the responses of the models with expected behavior of a physical network.

Model Verification

A series of tests were performed to analyze the logical behavior of each of the models under different traffic conditions. Additional tests were performed on specific components that were common to all of the models. These tests were not intended as a means of comparing the models; rather, they were performed to show that each of the models produced acceptable vehicle behavior.

Realism of Traffic Movement. A number of output statements were added to each of the programs. These statements provided detailed
information on every movement or decision of all vehicles in the system. Through an analysis of this output, the complete movement of each vehicle could be traced throughout its sojourn in the system. Three test networks were constructed as illustrated by Figure 14. Each model was run for 120 seconds of simulation time and the history of the vehicle movements was obtained. Cork boards, with drawings for each network, and map pins were used to visually trace each vehicle in the network. The use of this information resulted in several refinements to the programs.

Although these verification tests alone are not sufficient evidence to establish the credibility of the models, they do provide confidence in the basic sequence of actions followed by vehicles under different situations. Inoperative forms of the output statements used in this phase of testing have been retained in the program listings. A letter C and four asterisks precede the actual WRITE commands.

Pseudo-Random Number Tests. Although the method used by all models for generating pseudo-random numbers is considered acceptable, a series of tests were conducted to insure that proper parameters would be used in the actual runs. Utilizing different random number seeds the following procedure was employed. First 5000 pseudo-random numbers were generated. Next the numbers were grouped into classes with an interval width of 0.1 and a chi-square goodness-of-fit test performed. Then the sequence was subjected to a runs-up and runs-down test. Finally, serial correlation analysis for lags of 2, 4, 6 and 10 were used to test randomness.
Figure 14. Model Verification Networks

- a. Two-Lane by Two-Lane Intersection
- b. Four-Lane by Four-Lane Intersection
- c. Network of Two Two-Lane by Two-Lane Intersections
All tests were performed at a 5 per cent level of confidence. Only those seeds which passed all tests were retained for use in the actual experimental runs.

**Vehicle Generation Testing.** All models used the same method of generating exponentially distributed interarrival times. To test this generator, each seed selected by the previous tests was used to schedule arrivals for rates of 200, 300, 400, 500 and 600 vehicles per hour. For each volume, a five-hour sample was taken and the resulting interarrival time distribution was tested with a chi-square goodness-of-fit test. Additionally, five-minute traffic counts were obtained and tested against the appropriate Poisson distribution. All seeds passed each test at a 5 per cent level of significance.

**Model Validation**

After the logic of each program had been checked as described above, the comparison of model performance with expected behavior was undertaken. It is realized that a complete assessment of the accuracy of the simulation programs can be accomplished only through the application of an extensive field testing program. However, this type of validation is a most difficult undertaking even when unlimited resources are available. It was hoped that the models could be compared to some of the delay studies reported in the literature.

Berry (3) has published the results of extensive studies in the field measurement of delay. However, the published report on these studies is not complete in the description of the roadway sections and vehicle characteristics. Kell (24) encountered difficulty with the time
and manpower required to obtain field data, and with a more important limitation. The simulation technique which he used, like the ones used here, evaluates total system delay. Field studies are generally limited to measurement of stopped delay, ignoring delay accumulated in the process of accelerating and decelerating. The basic problem in validation is simply that information of the type readily obtained from the model is extremely difficult to measure in the field.

**Webster's Equation.** Webster (40) has developed a relationship between delay and saturation using a combination of theoretical and empirical techniques. The general delay equation is

\[
\begin{align*}
    d &= \frac{c(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - 0.65 \left( \frac{c}{q^2} \right)^{1/3} \times (2+5\lambda)
\end{align*}
\]

where

- \(d\) = average delay per vehicle on a particular approach arm of the intersection.
- \(c\) = cycle length in seconds.
- \(\lambda\) = portion of the cycle which is effectively green.
- \(q\) = flow rate in vehicles per second.
- \(x\) = the degree of saturation. This is the ratio of the actual flow to the maximal flow which can pass through this approach arm of the intersection.

The first term represents delay for traffic arriving at a uniform rate, while the second term represents added delay caused by the random nature of arrivals. The third term represents an empirical correction term added to improve the agreement of the equation with actual road conditions.
To test the validity of the simulation models, the delay estimates of all four models were compared to that computed by the Webster equation for various combinations of flow rates and turning maneuvers. The flow rates covered 200, 300, 400, 500 and 600 vehicles per hour. For each flow rate, the turn percentages indicated in Table 6 were specified.

Table 6. Turn Percentages in Validation Tests

<table>
<thead>
<tr>
<th>Flow Set</th>
<th>Right</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>II</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>III</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

All terms in the Webster equation are easily obtained with the exception of maximal flow. The maximal flow was determined in this study in accordance with the 1965 Highway Capacity Manual (20). The intersection approach considered was a two-lane, two-way street with no parking. Maximal flow was determined by assuming a load factor of unity and applying the appropriate turning maneuver correction factors.

Since the Highway Capacity Manual does not take into account opposing flow for left turns, an adjustment factor proposed by Collins (10) was used. In runs where left-turning traffic was present, the flow obtained from the Highway Capacity Manual was multiplied by $e^{-qt}(qt+1)$ where $q$ is the opposing flow in vehicles per second and $t$ is the average
acceptable gap in seconds. This adjusted maximal flow was then used to calculate average delay by the Webster equation.

Results of Tests. A typical two-lane, right-angled intersection was chosen for validation testing. The cycle length was set at 60 seconds with 32 seconds of green time on the main street. Main street flow rates covered 200, 300, 400, 500 and 600 vehicles per hour in each direction. The side street flow rate was held constant at five vehicles per hour. Three sets of flow rates were used in order to test the sensitivity of the models to the turn percentages indicated earlier.

Four runs of each simulation model were performed. Each run consisted of 30 minutes of simulation data time preceded by a 3-minute warm-up time. For each run the average delay per vehicle on each main approach link was computed and the two data points recorded. For each combination of model, flow rate and turning maneuver, the eight data points were averaged and a confidence interval computed. Table 7 gives the results of the simulation model comparisons to the delay time computed by the Webster equation. The adjusted equation, suggested by Collins, is also included for reference. However, all statistical comparisons were performed with the more accepted Webster equation. Figures 15, 16 and 17 present a graphical representation of these data.

The overall agreement between the model results and the Webster equation are satisfactory. The CONT model and UB model produced only one rejection each out of 15 independent applications of a confidence interval test. The results of these two models also compare favorably in the low and medium flow ranges. A wider dispersion is noted in the
500 to 600 vehicles per hour range. The ZONE model is significantly different from the Webster equation at four points. A general trend at underestimating delay in all but the high flow rates can be observed from the graphs of the data. The NEXT model was also found to be significantly different from the Webster equation at four points. In general this model tended to overestimate delay. A more favorable agreement was found between the NEXT model and the adjusted equation proposed by Collins.

Table 7. Average Delay per Vehicle for Validation Study

<table>
<thead>
<tr>
<th>FLOW</th>
<th>CONT</th>
<th>UB</th>
<th>ZONE</th>
<th>NEXT</th>
<th>Webster</th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>9.36</td>
<td>8.81</td>
<td>8.70</td>
<td>9.99</td>
<td>9.6</td>
<td>9.6</td>
</tr>
<tr>
<td>300</td>
<td>11.13</td>
<td>11.03</td>
<td>10.05</td>
<td>12.47</td>
<td>10.9</td>
<td>11.2</td>
</tr>
<tr>
<td>400</td>
<td>12.20</td>
<td>13.39</td>
<td>11.41</td>
<td>15.42</td>
<td>12.6</td>
<td>13.5</td>
</tr>
<tr>
<td>500</td>
<td>15.53</td>
<td>18.55</td>
<td>17.09</td>
<td>19.89*</td>
<td>15.0</td>
<td>18.4</td>
</tr>
<tr>
<td>600</td>
<td>21.77</td>
<td>53.36*</td>
<td>35.07*</td>
<td>34.07</td>
<td>20.6</td>
<td>77.8</td>
</tr>
</tbody>
</table>

*Significant at a 5 per cent level of confidence.

Figure 15. Comparison of Webster Equation to Predictions from Models for No Turns
Figure 16. Comparison of Webster Equation to Predictions from Models for 5% Left-5% Right Turns
Figure 17. Comparison of Webster Equation to Predictions from Models for 10% Left-10% Right Turns
Network Performance Comparison

After confidence had been established in the performance of the individual models, a series of computer runs were undertaken whereby the performance of the models could be further analyzed and compared. Various service volumes as well as network configurations were considered in order to ascertain their influence on model behavior.

Description of Test Networks

To test the effects of network configuration on model performance, it was decided to run each model on three different street patterns—a single intersection, an arterial pattern and a closed network.

Network Considered. To approximate an urban situation, each intersection was assumed to be controlled by a fixed time traffic signal and all link lengths were less than 800 feet. A link-node diagram of each network configuration is given in Figure 18. The numbers in parentheses are the link lengths. All streets are four lanes wide with no curbside parking permitted.

To facilitate comparing the performance of the models on different configurations, the three networks are nested; the single intersection is contained in the arterial pattern which is part of the closed network. Since the input links to the intersection at Third Street and Main Street (see Figure 18) have the same characteristics in all configurations, the performance measures for these links formed the basis for model comparison. In order to compare delay predictions from the three configuration, it was essential that some common street segment be included in each configuration. The intersection of Main Street and
Figure 18. Link-Node Diagrams of Test Networks
Third Street forms such a base.

**Selection of Signal Settings.** In order to insure that the signal settings were not a major source of delay, a progressive timing pattern was determined for each of the arterial flows. Without progression, the capacity of an individual intersection may be decreased because vehicles may be unavoidably detained due to poor sequencing of the signals. With proper progression, the main platoon will witness delays only when starting at the first signal. If the progression speed is maintained, these vehicles can then pass through the other signals without further delay.

To determine the progressive timing patterns, the algorithm proposed by Morgan and Little (30) was programmed. This algorithm determines the maximal equal bandwidths* for opposing flows on a two-way arterial. The program was run for each arterial street for signal cycle times ranging between 60 and 90 seconds. Since the 90-second cycle produced the largest bands on each artery, it was selected as the network cycle length. Figure 19 illustrates the resulting time-space diagram for the three main streets in the connected network.

**Volume Considerations.** For the purposes of the simulation input, traffic volumes are specified at the external nodes only. Internal volumes are obtained as a result of the simulation process.

A procedure was established to insure equivalent volumes, for all network configurations, on the four links leading to the main

---

* The bandwidth is defined as the size (in seconds) of the largest platoon which can pass through a series of signals without stopping.
Figure 19. Time-Space Diagrams for 90-Second Cycle

a. East Side Drive Progression

b. Main Street Progression

c. West Side Drive Progression
intersection. Let

\[ p_{11}x_1 + p_{12}x_2 + \ldots + p_{1n}x_n = V_1 \]
\[ p_{21}x_1 + p_{22}x_2 + \ldots + p_{2n}x_n = V_2 \]
\[ \vdots \]
\[ p_{n1}x_1 + p_{n2}x_2 + \ldots + p_{nn}x_n = V_n \]

where \( x_i \) is the volume per hour on the \( i \)th link,

\[ P_{ii} = \begin{cases} +1 & \text{if the link is a source link, or} \\ -1 & \text{if the link is an internal link or a sink link,} \end{cases} \]

\( p_{ij} \) is the probability of going to the \( i \)th link from the \( j \)th link \( i \neq j \),

\( V_i \) is the external source volume for the \( i \)th link, and

\( n \) is the number of links.

For any given set of external volumes \( \{V_i\} \) the internal volumes for each link \( \{x_i\} \) can be calculated by solving the simultaneous equations.

To arrive at three different flow levels for the various network configurations, a number of external source volume combinations were considered for the closed network. The simultaneous equations for each volume combination were solved and three combinations were selected which yielded light, medium and heavy service volumes for the principal intersection. Next the source links for the arterial pattern and the single intersection were considered. The theoretical volumes \( \{x_i\} \) determined in the closed network were used as source volumes in the
smaller networks. Thus, the four links of importance theoretically are being subjected to the same volumes of traffic for each network configuration. Appendix F lists the volumes projected for the closed network. It should be noted that the heavy traffic has been chosen at a level below 500 vehicles per hour per lane in order to insure stability in the network flow.

**Test Results**

Two computer runs were made for each combination of simulation model, network configuration and flow rate. Each run consisted of 45 minutes of simulated data gathering time. The warm-up times were dependent on network configuration as given in Table 8.

<table>
<thead>
<tr>
<th>Network Configuration</th>
<th>Warm-up Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersection</td>
<td>5</td>
</tr>
<tr>
<td>Arterial</td>
<td>10</td>
</tr>
<tr>
<td>Network</td>
<td>15</td>
</tr>
</tbody>
</table>

**Model Performance.** During each run, the average delay per vehicle for the main intersection was computed and recorded. The results of the test runs are given in Table 9. A three-factor analysis of variance of these data is presently in Table 10. All effects are considered as fixed effects and the assumption of an analysis of variance test appear to apply to this experiment.
Table 9. Average Delay per Vehicle at Main Intersection of Test Networks

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Flow</th>
<th>MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CONT</td>
</tr>
<tr>
<td><strong>Intersection</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td></td>
<td>16.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.53</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>21.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20.96</td>
</tr>
<tr>
<td>Heavy</td>
<td></td>
<td>31.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>46.10</td>
</tr>
<tr>
<td><strong>Arterial</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td></td>
<td>15.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.87</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>18.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19.05</td>
</tr>
<tr>
<td>Heavy</td>
<td></td>
<td>31.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29.74</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td></td>
<td>12.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.13</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>17.67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14.70</td>
</tr>
<tr>
<td>Heavy</td>
<td></td>
<td>35.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25.03</td>
</tr>
</tbody>
</table>
Table 10. Analysis of Variance of the Model Comparison Data

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models ($M_i$)</td>
<td>557.49</td>
<td>3</td>
<td>185.83**</td>
</tr>
<tr>
<td>Networks ($N_j$)</td>
<td>1030.27</td>
<td>2</td>
<td>515.13**</td>
</tr>
<tr>
<td>Flows ($F_k$)</td>
<td>3481.75</td>
<td>2</td>
<td>1740.87**</td>
</tr>
<tr>
<td>$M \times N$</td>
<td>54.70</td>
<td>6</td>
<td>9.12</td>
</tr>
<tr>
<td>$N \times F$</td>
<td>331.52</td>
<td>6</td>
<td>55.25**</td>
</tr>
<tr>
<td>$N \times F$</td>
<td>28.79</td>
<td>4</td>
<td>7.20</td>
</tr>
<tr>
<td>$M \times N \times F$</td>
<td>37.68</td>
<td>12</td>
<td>3.14</td>
</tr>
<tr>
<td>Error</td>
<td>423.98</td>
<td>36</td>
<td>11.78</td>
</tr>
<tr>
<td>TOTAL</td>
<td>5946.19</td>
<td>71</td>
<td></td>
</tr>
</tbody>
</table>

**Significant at 1%.

It can be seen that all main effects are highly significant. With respect to the first factor, this implies that the type of model used does influence the delay prediction. The data, presented in Table 9, indicate that the ZONE and NEXT models predicted delays below that of the CONT and UB models for all combinations of flow and configuration except the light flow on the single intersection. The validation test somewhat indicated this possibility for the ZONE model; however, the NEXT model predicted higher delays than expected in single-lane tests.

The assumptions regarding vehicle interactions and accelerations are thought to be a primary reason for these low estimates of delay. In the ZONE model vehicles are either moving at "free speed" or standing still. Such an assumption requires instantaneous acceleration to free speed. Furthermore, there is no interaction between vehicles advancing
in the same direction. The NEXT model does not take into account any vehicle interaction on the link during the vehicle's journey to the queue state. Furthermore, acceleration delay considerations, due to a low initial speed, are minimal when the vehicle is scheduled to travel down the link. This latter problem would not affect delay calculations in modeling a single intersection, since only incoming links to the intersection are significant and initial speeds of new vehicles are set equal to their desired speeds.

Predictions of delay from the UB model usually fall between those from the CONT model and the ZONE and NEXT models. The CONT model predicted higher delays than the other models in all but one combination of configuration and flow.

The effect of network configuration on vehicle delay was significant. This result is reasonable when the advantages of the progressive signal system are considered. Vehicles, which enter the "green band" at an exterior intersection, could pass through the system without delay. Other vehicles, which should have stopped at the main light, were delayed at a previous intersection and forced to wait for the band. As configuration was changed from an arterial roadway to the closed network, the side street progression further reduced delay.

As expected, flow rates had a highly significant effect on delay. As flow increased, left-turning vehicles found smaller gaps and were forced to wait, queues were longer and some vehicles were forced to wait through more than a partial cycle of the light. Since average link speed is reduced, vehicles can no longer stay in the green band and hence some
of the advantages of the progressive signal system are lost.

A highly significant interaction was found between models and flows. This result again agrees with the graphical results obtained in the validation testing. Certain models were more sensitive to changes in flow rate than other models. No other interactions were statistically significant.

In addition to the above analysis, each model was subjected to a single long simulation run in order to obtain a time series of the average delay occurring on specific links of the network. The intersection configuration under medium flow was selected for this test. A five-hour sample run was made. The North and South approaches to the main intersection were chosen as the links of interest. Every 90 seconds, the delay per vehicle exiting the link was computed and recorded. The average delay encountered in each time interval was computed and the results were punched on computer cards for further analysis.

The sequence of observations \( \{d_j\} \) for a link constitutes a time series of 200 observations. The first ten observations were discarded to allow for system loading. An analysis of the autocorrelation of the series was then conducted to examine the dependence of delay at time \( t \) with some previous delay. If a higher than average observed delay tends to be followed by another higher than average delay observed \( k \) time periods later, the autocorrelation between \( d_t \) and \( d_{t+k} \) is positive. Similarly, if a lower than average delay is followed by another lower than average delay at a lag of \( k \) time periods, the autocorrelation between \( d_t \) and \( d_{t+k} \) is negative. An estimate of the autocorrelation for
A lag of \( k \) time units can be obtained from the sample autocorrelation function

\[
\hat{\rho}_k = \frac{1}{N-k} \sum_{t=1}^{N-k} (d_t - \bar{d})(d_{t+k} - \bar{d})
\]

\[
= \frac{\frac{1}{N} \sum_{t=1}^{N} (d_t - \bar{d})^2}{\frac{1}{N} \sum_{t=1}^{N} (d_t - \bar{d})^2}
\]

\( k = 0, 1, \ldots, K \)

where \( N \) is the total number of observations.

\( k \) is the lag size.

\( d_t \) is the 90-second average delay observed at period \( t \).

\( K \) is some arbitrary upper limit on lag, usually \( \leq \frac{N}{4} \).

\( \bar{d} \) is the average of the \( d_t \)'s.

The partial autocorrelation or conditional correlation function is also estimated using the recursive method outlined in Box and Jenkins (6). This function is also helpful in identifying the type of dependencies which exist.

Typical plots of a sample autocorrelation function and the associated partial autocorrelation function are presented in Figures 20 and 21 for the North approach link and South approach link, respectively. The sample autocorrelation functions for all four models are given in Appendix G.

The first observation that should be made is that the different models produced very similar plots for the two links. This indicates that the form of the statistical model used to estimate delay in one model could be fitted to the observed delay found in a different model.
Figure 20. Sample Autocorrelation Function and Partial Autocorrelation Function for North Approach Link
Figure 21. Sample Autocorrelation Function and Partial Autocorrelation Function for South Approach Link
by using different estimates for the parameters of the statistical model. That is to say, all models seem to exhibit the same interrelationships in period-by-period or cycle-by-cycle delay.

The second observation is that the two links exhibit different autocorrelation functions. The North approach plot suggests that the delay time during period $t$ has a positive correlation with period $t - 1$ and all other relationships are zero. The South approach plots suggest independence between each period. Although the link flow volumes for these two links are about the same, a noticeable difference exists in the turning percentages. Five per cent of the vehicles on the North approach will turn left, while no left turns are permitted on the South approach. The existence of left-turn delays appears to be sufficient to cause the appearance of a first order autocorrelation.

**Computer Requirements.** The program execution time was recorded for each of the test runs and is given in Table 11. The ZONE model was the fastest in all cases. The NEXT model was very fast for simple intersection work but quickly lost any computational advantage it possessed as the network complexity increased. The UB model was the slowest of the models tested. Although the vehicle movement concepts in the UB model are not as complex as the car-following restrictions of the CONT model, the repeated checking of each block and the complex routines for intersection movement require greater computer time to process. The algebraic formulas needed for car-following restrictions are well suited for computer programming.

The computer memory requirements of the different models is important if large networks are being considered. The actual requirements
Table 11. Program Execution Time for Test Runs (in Seconds)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Flow</th>
<th>CONT</th>
<th>UB</th>
<th>ZONE</th>
<th>NEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intersection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>17.0</td>
<td>18.0</td>
<td>38.4</td>
<td>3.4</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td>18.0</td>
<td>18.0</td>
<td>38.1</td>
<td>3.3</td>
<td>6.0</td>
</tr>
<tr>
<td>Medium</td>
<td>26.0</td>
<td>26.0</td>
<td>47.0</td>
<td>4.3</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>26.0</td>
<td>26.0</td>
<td>48.1</td>
<td>4.2</td>
<td>9.0</td>
</tr>
<tr>
<td>Heavy</td>
<td>49.0</td>
<td>53.0</td>
<td>66.2</td>
<td>4.3</td>
<td>29.0</td>
</tr>
<tr>
<td></td>
<td>53.0</td>
<td>78.4</td>
<td>5.0</td>
<td></td>
<td>16.2</td>
</tr>
<tr>
<td><strong>Arterial</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>77.2</td>
<td>81.2</td>
<td>139.2</td>
<td>14.0</td>
<td>32.4</td>
</tr>
<tr>
<td></td>
<td>81.2</td>
<td>136.0</td>
<td>15.0</td>
<td></td>
<td>33.1</td>
</tr>
<tr>
<td>Medium</td>
<td>120.4</td>
<td>114.0</td>
<td>182.5</td>
<td>20.0</td>
<td>76.4</td>
</tr>
<tr>
<td></td>
<td>114.0</td>
<td>170.5</td>
<td>13.4</td>
<td></td>
<td>76.2</td>
</tr>
<tr>
<td>Heavy</td>
<td>202.0</td>
<td>179.1</td>
<td>289.0</td>
<td>17.0</td>
<td>168.1</td>
</tr>
<tr>
<td></td>
<td>179.1</td>
<td>267.0</td>
<td>18.0</td>
<td></td>
<td>163.4</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>221.1</td>
<td>207.0</td>
<td>340.0</td>
<td>36.0</td>
<td>199.3</td>
</tr>
<tr>
<td></td>
<td>207.0</td>
<td>355.2</td>
<td>35.5</td>
<td></td>
<td>166.0</td>
</tr>
<tr>
<td>Medium</td>
<td>334.3</td>
<td>316.0</td>
<td>462.0</td>
<td>42.4</td>
<td>333.0</td>
</tr>
<tr>
<td></td>
<td>316.0</td>
<td>482.0</td>
<td>36.0</td>
<td></td>
<td>318.0</td>
</tr>
<tr>
<td>Heavy</td>
<td>577.0</td>
<td>483.0</td>
<td>729.0</td>
<td>46.4</td>
<td>861.3</td>
</tr>
<tr>
<td></td>
<td>483.0</td>
<td>923.0</td>
<td>46.1</td>
<td></td>
<td>656.0</td>
</tr>
</tbody>
</table>
can be estimated by using the information given in Table 12.

Table 12. Computer Memory Requirements

<table>
<thead>
<tr>
<th>Model</th>
<th>Base Requirement</th>
<th>Per Node</th>
<th>Per Signal</th>
<th>Per Vehicle</th>
<th>Per Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONT</td>
<td>33,000</td>
<td>15</td>
<td>15</td>
<td>12</td>
<td>33</td>
</tr>
<tr>
<td>UB#</td>
<td>26,000</td>
<td>17</td>
<td>15</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td>ZONE**</td>
<td>17,000</td>
<td>12</td>
<td>15</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>NEXT</td>
<td>20,500</td>
<td>15</td>
<td>15</td>
<td>8</td>
<td>37</td>
</tr>
</tbody>
</table>

Additional Requirements
* The UB model requires one word of computer memory for each unit block.
** The ZONE model requires one word of computer memory for each zone.

In order to reduce the amount of computer memory necessary, an attempt was made to code information as illustrated in Figure 22.

![Figure 22. Basic Concept of Coded Data Words](image)

The UB model was selected. Vehicle data were compressed to three computer words and unit blocks were packed three to a word. However, resulting test runs increased running time by a factor of six and the coded model was abandoned.
Secondary Considerations

In addition to the analysis of the effects of modeling strategies, a limited number of secondary problems were investigated.

Scan Time Effects

Zone models reported in the literature (16,33,39) have used scan times of 2, 4 and 5 seconds. To test the effect of scan time on vehicle delay, a series of test runs of the ZONE model was conducted with scan times of 2, 3, 4 and 5 seconds. Since the basic modeling concept was introduced for the purpose of modeling a system larger than a single intersection, the arterial configuration was selected as the test network. Five replications of light, medium and heavy flow rates were run for each scan time. The average system delay per vehicle per link encountered was computed for each run, and the results are presented in Table 13. A two-way analysis of variance was performed on these data, and the results are given in Table 14. Both main effects and their interaction are highly significant. A graphic illustration of these effects is given in Figure 23. To further analyze the interaction of these two factors the Duncan Multiple Range test was performed on the data by individually testing each flow condition. At the low level no significant differences were exhibited between scan time results. At the medium flow rate, two pairs of predictions were significant at a 5% level of confidence: the 3 second-2 second difference and the 3 second-4 second difference. At the heavy flow rate, all pairs of differences were significant except the 2 second-4 second combination.
Table 13. Effects of Scan Time on ZONE Model Response Predictions

<table>
<thead>
<tr>
<th>Flow</th>
<th>Time Step (Seconds)</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.74</td>
<td>14.91</td>
<td>12.95</td>
<td>14.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.05</td>
<td>15.91</td>
<td>13.92</td>
<td>15.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.25</td>
<td>15.85</td>
<td>13.22</td>
<td>14.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.38</td>
<td>15.65</td>
<td>13.56</td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.74</td>
<td>16.65</td>
<td>13.10</td>
<td>14.86</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>16.15</td>
<td>24.84</td>
<td>16.24</td>
<td>18.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.46</td>
<td>29.25</td>
<td>16.55</td>
<td>21.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.22</td>
<td>25.99</td>
<td>16.38</td>
<td>23.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.71</td>
<td>24.33</td>
<td>16.01</td>
<td>20.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.67</td>
<td>31.45</td>
<td>16.94</td>
<td>21.40</td>
</tr>
<tr>
<td>Heavy</td>
<td></td>
<td>26.34</td>
<td>76.63</td>
<td>23.60</td>
<td>44.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.74</td>
<td>71.60</td>
<td>28.67</td>
<td>50.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25.15</td>
<td>81.37</td>
<td>24.28</td>
<td>69.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23.60</td>
<td>74.93</td>
<td>21.59</td>
<td>56.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.92</td>
<td>91.63</td>
<td>29.48</td>
<td>78.45</td>
</tr>
</tbody>
</table>

Table 14. Analysis of Variance of the Effects of Scan Time on the ZONE Model

<table>
<thead>
<tr>
<th>Source of Variance</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan Times (T_i)</td>
<td>5268.66</td>
<td>3</td>
<td>1756.22**</td>
</tr>
<tr>
<td>Flows (F_j)</td>
<td>12911.04</td>
<td>2</td>
<td>6455.52**</td>
</tr>
<tr>
<td>T x F</td>
<td>5449.60</td>
<td>6</td>
<td>908.27**</td>
</tr>
<tr>
<td>Error</td>
<td>1159.07</td>
<td>48</td>
<td>24.15</td>
</tr>
<tr>
<td>TOTAL</td>
<td>24788.37</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

**Significant at 1%.**
Figure 23. Scan Time-Flow Interaction for ZONE Model
The queue discharge mechanism is thought to be the underlying cause of this behavior. The number of vehicles which can depart a queue is the scan time divided by two seconds per vehicle. Since only an integer number of vehicles may depart the queue, the two- and three-second scan times permit one departure each scan while the four- and five-second scan times permit two discharges. The time between departures is then equal to the desired two seconds for the two- and four-second scan times. The three-second scan is forced to a three-second time between departures, while the average time between departures for the five-second scan is 2-1/2 seconds. The longer departure time yields more queue congestion.

The NEXT model was also tested for step size effects. The scale factor which had been used to force integer clock units was relaxed and the model was permitted to assume values in tenths of seconds. Five replications of three arterial flow rates were submitted for each step size. The data is given in Table 15. Table 16 is the analysis of variance of the data.

Both main effects are highly significant. The interaction between clock units and flows is also significant. Figure 24 illustrates this interaction.

The queue discharge process is again considered to be a contributing factor to this interaction. The smaller clock units allow a closer approximation of the discharge times presented in the model. Integer clock units cause the model to discard the fractional part of the time between discharges, thus causing a more rapid discharge
Table 15. Effects of Clock Units on NEXT Model Response Predictions

<table>
<thead>
<tr>
<th>Flow</th>
<th>Clock Unit (Seconds)</th>
<th>1</th>
<th>1/10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td>11.09</td>
<td>11.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.08</td>
<td>12.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.39</td>
<td>12.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.34</td>
<td>11.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.97</td>
<td>13.44</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>13.36</td>
<td>13.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.02</td>
<td>15.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.06</td>
<td>13.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.97</td>
<td>13.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13.66</td>
<td>14.18</td>
</tr>
<tr>
<td>Heavy</td>
<td></td>
<td>19.53</td>
<td>29.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19.57</td>
<td>32.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.26</td>
<td>20.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.76</td>
<td>19.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22.08</td>
<td>25.84</td>
</tr>
</tbody>
</table>

Table 16. Analysis of Variance of Clock Unit Effects on NEXT Model Response Predictions

<table>
<thead>
<tr>
<th>Source of Variance</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Units (C_i)</td>
<td>51.17</td>
<td>1</td>
<td>51.17**</td>
</tr>
<tr>
<td>Flows (F_j)</td>
<td>588.50</td>
<td>2</td>
<td>294.25**</td>
</tr>
<tr>
<td>(C \times F)</td>
<td>64.84</td>
<td>2</td>
<td>32.42*</td>
</tr>
<tr>
<td>Error</td>
<td>154.56</td>
<td>24</td>
<td>6.44</td>
</tr>
<tr>
<td>TOTAL</td>
<td>859.07</td>
<td>29</td>
<td></td>
</tr>
</tbody>
</table>

*Significant at 5%.
**Significant at 1%.
Figure 24. Interaction Between Clock Unit and Flow for NEXT Model
process. As flow increases and the average queue lengths increase, a reduction in delay time is seen.

Network Preloading

Subroutines were written for the CONT and UB models which would calculate the expected internal flow rates for each link of the network. Vehicles were then created and placed on the links before the simulation started. The models were then allowed a five-minute transition time to remove any effects of this starting condition. The running time of the CONT model was 83 seconds faster than the average time indicated for the heavy network configuration. The preloaded UB model bettered the previous results by 123 seconds. Although these results are within the range of random error, it was noted that preload calculations consumed only 3.5 seconds of computer time. Thus it is advisable to consider preloading, if large networks are simulated with the more complex models.

Interarrival Time Distributions

Since much of the literature on traffic simulation is concerned with interarrival distributions, it was decided to test the effects of the assumed distribution. The NEXT model was revised to permit the simulation of interarrival times which followed an Erlang distribution with the identical mean and with $k$ equal to two. Five replications of two configurations and three flow volumes were included in the test.

The average delay per vehicle at the main intersection is reported in Table 17. The analysis of variance of these data is given in Table 18.
Table 17. Effects of Interarrival Time Distribution on the NEXT Model Response Distribution

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Flow</th>
<th>Interarrival Time Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Erlang</td>
</tr>
<tr>
<td>Low Intersection</td>
<td>16.03</td>
<td>17.07</td>
</tr>
<tr>
<td></td>
<td>15.70</td>
<td>16.95</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td>16.41</td>
</tr>
<tr>
<td></td>
<td>15.99</td>
<td>16.80</td>
</tr>
<tr>
<td></td>
<td>16.93</td>
<td>17.22</td>
</tr>
<tr>
<td>Medium</td>
<td>18.55</td>
<td>17.85</td>
</tr>
<tr>
<td></td>
<td>18.26</td>
<td>18.85</td>
</tr>
<tr>
<td></td>
<td>17.65</td>
<td>18.68</td>
</tr>
<tr>
<td></td>
<td>18.03</td>
<td>18.24</td>
</tr>
<tr>
<td></td>
<td>20.54</td>
<td>19.66</td>
</tr>
<tr>
<td>High Intersection</td>
<td>23.68</td>
<td>28.60</td>
</tr>
<tr>
<td></td>
<td>25.02</td>
<td>24.59</td>
</tr>
<tr>
<td></td>
<td>24.85</td>
<td>27.03</td>
</tr>
<tr>
<td></td>
<td>28.70</td>
<td>24.64</td>
</tr>
<tr>
<td></td>
<td>26.72</td>
<td>24.89</td>
</tr>
<tr>
<td>Low Artery</td>
<td>11.06</td>
<td>11.09</td>
</tr>
<tr>
<td></td>
<td>12.34</td>
<td>12.08</td>
</tr>
<tr>
<td></td>
<td>11.29</td>
<td>11.39</td>
</tr>
<tr>
<td></td>
<td>11.58</td>
<td>11.34</td>
</tr>
<tr>
<td></td>
<td>12.18</td>
<td>12.97</td>
</tr>
<tr>
<td>Medium Artery</td>
<td>13.08</td>
<td>13.36</td>
</tr>
<tr>
<td></td>
<td>12.76</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>13.38</td>
<td>13.06</td>
</tr>
<tr>
<td></td>
<td>12.44</td>
<td>12.97</td>
</tr>
<tr>
<td></td>
<td>13.20</td>
<td>13.66</td>
</tr>
<tr>
<td>High Artery</td>
<td>17.39</td>
<td>19.53</td>
</tr>
<tr>
<td></td>
<td>17.05</td>
<td>19.57</td>
</tr>
<tr>
<td></td>
<td>16.66</td>
<td>17.26</td>
</tr>
<tr>
<td></td>
<td>19.72</td>
<td>15.76</td>
</tr>
<tr>
<td></td>
<td>23.37</td>
<td>22.08</td>
</tr>
</tbody>
</table>
Table 18. Analysis of Variance of Effects of Interarrival Time Distribution on NEXT Model

<table>
<thead>
<tr>
<th>Source of Variance</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributions (D₁)</td>
<td>1.17</td>
<td>1</td>
<td>1.17</td>
</tr>
<tr>
<td>Networks (N_j)</td>
<td>490.09</td>
<td>1</td>
<td>490.09**</td>
</tr>
<tr>
<td>Flows (F_k)</td>
<td>746.47</td>
<td>2</td>
<td>373.24**</td>
</tr>
<tr>
<td>D × N</td>
<td>.02</td>
<td>1</td>
<td>.02</td>
</tr>
<tr>
<td>D × F</td>
<td>.33</td>
<td>2</td>
<td>.16</td>
</tr>
<tr>
<td>N × F</td>
<td>13.69</td>
<td>2</td>
<td>6.84*</td>
</tr>
<tr>
<td>D × N × F</td>
<td>1.01</td>
<td>2</td>
<td>.51</td>
</tr>
<tr>
<td>Error</td>
<td>98.22</td>
<td>48</td>
<td>2.05</td>
</tr>
<tr>
<td>TOTAL</td>
<td>1351.01</td>
<td>59</td>
<td></td>
</tr>
</tbody>
</table>

*Significant at 5%.
**Significant at 1%.

The analysis of variance indicates that there is no significant effect in model performance caused by changing the arrival time distribution. Flow and configuration are highly significant. It is also noted that the interaction between flow and configuration is significant. This interaction was not significant in the analysis of variance for all four models. The use of five replications per cell may have unmasked an unknown interaction.
CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Summary

Four computer programs representing different strategies for simulating urban vehicular flow were programmed and tested. A simple two-lane, right-angled intersection was simulated under combinations of varying flow rates and turning probabilities and the resulting delay prediction compared to Webster's equation for average delay.

A series of tests were then conducted to determine the significance of model type, network configuration and flow rate on the prediction of average vehicle delay. A second set of tests were performed to examine the dynamic properties of delay predictions over extended periods of time.

A number of secondary considerations in traffic simulation were examined. First two of the models were selected and the effects of simulation scanning time were examined. Next, a method of preloading the network with vehicles was devised and the reduction in computation time of preloading was compared to the times obtained by starting the networks in an empty state. Finally, one of the models was selected to test the significance of the interarrival time distribution. Model performance using exponentially distributed interarrival times was compared to performance under the assumption of an Erlang distribution.
Conclusions

Effects of Modeling Philosophies on Performance

There is a statistically significant difference in the delay predictions of the models. Although the time dependent sequence of delays obtained from the various models appears to follow the same pattern, the magnitudes of the delay predicted by the models differs significantly.

Model running times also exhibit a dependence on model type. The ZONE model times were 10 to 15 times faster than the more complex UB and CONT models. Differences in computer memory requirements were not as significant as differences in running speeds, and it was found that this parameter was highly dependent on the actual network being modeled.

Warm-Up Time and Preloading Considerations

The results regarding warm-up time were inconclusive. The correlation between delays observed in different time periods appears to be restricted to a one-period (signal cycle time) lag. However, it is noted that this factor depends on turning probabilities and further research is needed to examine other contributing factors.

The magnitude of the savings in computational time of preloading a network with vehicles was not statistically significant, based on the limited sample size. However, the short time necessary to preload the network indicates that a saving should result from preloading larger networks.
Desirable Step Size

Both the step size of the ZONE model and the unit size of the clock for the NEXT model were shown to have a significant effect on delay prediction. Since field data from actual network delays were not available, no specific step size was singled out as being best. Extensive field validation is necessary to fix this parameter.

Interarrival Time Distribution

No significant effect was determined in the choice of an interarrival time distribution. The computational complexity of the logarithmic calculations in the exponential distribution and the Erlang distribution suggest that possibly some other distribution might be more desirable.

Recommendations

Direction of Future Traffic Models

This research has only indicated that a difference exists in the responses produced by different model types. An extensive field validation procedure is necessary before the best model can be identified. This research is greatly needed due to the increasing number of applications of traffic simulation.

The unit block type models do not appear to offer any significant advantages. Large computer memory requirements, slow running speeds and complex programming routines suggest that they should be removed from serious considerations for network modeling.

The tremendous speed advantage of the ZONE model suggests that further research in this modeling technique should be conducted to
improve the performance prediction capabilities of this model. Care must be taken in establishing the desirable step size.

It is the conjecture of this researcher that the NEXT model can be improved to obtain the speed advantage in network situations which was exhibited in the intersection studies. Irregardless, the chain structure used to control infrequent events, such as signal changes, should be employed by all model types. Repeat checking of the status of a green light can only lead to wasted computer processing time. The importance of this factor increases as smaller step sizes are considered.

**Model Validation Practices**

There is no substitute for the validation of model response to field observations. This validation must contrast the prediction of the performance measures with actual system characteristics under similar situations. If the models are to be used to measure network characteristics, then actual networks must be used for field observations.

Before field validation begins, the method of gathering field data needs to be reviewed and adjusted for network consideration. Methods must be devised for measuring the performance measures in the field as well as in the simulation models. Present techniques are lacking in this area.

**Performance Measures.** Some research is needed in the area of performance measure definition and identification. In many cases, models have been developed and the performance measures readily available used to contrast system performance. Significant measures need to be identified and then operationally defined.
The measurement and analysis of simulation responses requires more investigation. The simple application of autocorrelation analysis presented in Chapter 4 is but one of a multitude of analytical techniques available to examine the time dependent system responses. Extensive research is needed in this area to analyze both the system and the simulation responses.
APPENDICES
APPENDIX A

INPUT FORMAT SPECIFICATIONS

Each input deck consists of a run specification card and a network data deck. The network data deck has the same format for all four models. The run specification card is the same for the CONT, UB and NEXT models. One additional piece of information is required for the ZONE model.

Run Specification Card

Each input deck must begin with a run specification card. A free field format is used for this card and all quantities are read as integer variables. For the CONT, UB and NEXT models the following data must be entered:

1. Run Index
2. Network Index
3. Amber Time (Seconds)
4. Random Number Seed
5. Data Running Time (Minutes)
6. Warm-Up Time (Minutes)

The ZONE model requires:

1. Run Index
2. Network Index
3. Amber Time (Seconds)
4. Random Number Seed
5. Scan Time Step Size (Seconds)
6. Data Running Time (Minutes)
7. Warm-Up Time (Minutes)
Network Data Deck

The network data deck is divided into three sections. First the data cards on all external nodes are read. Next, pairs of cards are read for each internal node. The first card specifies data characteristics of the node while the second card identifies the signal characteristics. Finally, the data cards on all links are read. Each section of cards is followed by a blank card as a section delimiter.

Each external node in the network must be identified by one card as specified below:

Column
1  The card identification code 1 must be entered (I1).
2-4 The node identification number must be entered (I3).
5-6 The node type must be entered (I2).
   0 - Both generate and terminate.
   1 - Generate only.
   2 - Terminate only.
10-14 The node generation volume in vehicles per hour (F5.0).
20  The number of lanes per link leading from the node (I1).

Each internal node will have two data cards associated with it. The first card identifies the node characteristics and the format is:

Column
1  The card identification code 2 must be entered (I1).
2-4 The node identification number must be entered (I3).
5-6 The node type must be entered (I2).
   5 - Fixed time signal controlled.
7-9  The identification number of the nodes connected to the node.
10-12 A counter-clockwise sequence is assumed starting with one of the major street approach nodes.
13-15
The node geometry (II)
1 - Two-lane by two-lane.
3 - Four-lane by four-lane.

The second internal node data card identifies signal control parameters and is specified as follows:

**Column**

1    The card identification code 3 must be entered (II).
2-4  The node identification number must be entered (I3).
5-7  The signal cycle time in seconds (I3).
8-10 The major street offset time* in seconds (I3).
11-13 The major street green time in seconds (I3).
14-16 The minor street offset time in seconds (I3).
17-19 The minor street green time in seconds (I3).

Each network link requires one link data card in the network data deck. The format specification is given below:

**Column**

1    The card identification code 4 must be entered (II).
2-4  The identification number of the source node must be entered (I3).
5-7  The identification number of the head node must be entered (I3).
8-11 The link length in feet (I4).
12   The number of lanes (II).
13-15 The source node identification number of the link opposing left-turning traffic (I3).
16-18 The probability of a left turn (in parts per thousand) at the head node (F3.3).
19-21 The probability of a straight movement at the head node (F3.3).
22-24 The head node identification numbers of the destination links for left-, straighting and right-turning traffic.
25-27 The probability of a straight movement at the head node (F3.3).
28-30

* Here offset time is defined to be the number of seconds until the first red phase.
APPENDIX B

FORTRAN LISTING OF CONT MODEL
DEFINS PROCEDURE

***** SETUP FOR NODES.

PARAMETER MAXNO=35
REAL NOPARS, NOCLK.
COMMON /NODES/ NO1NO(MAXNO), NOTYPE(MAXNO), NONXT(MAXNO),
1 NOPARS(MAXNO), NOINP(MAXNO), NOOUTP(MAXNO),
2 NOSEED(MAXNO), NOSIG(MAXNO), NOCLK(MAXNO), NUMNOS

***** SETUP FOR SIGNALS.

PARAMETER MAXSIG=20
INTEGER AMBERT, SINGNP, SIGCYC, SIGSTA, SIGOFF, SIGRED, SIGCLK
COMMON /SIG/ SIGCYC(MAXSIG), SIGSTA(MAXSIG), SIGOFF(MAXSIG),
1 SIGRED(MAXSIG), SIGCLK(MAXSIG),
2 SIGNP(MAXSIG), NUMSIG, AMBERT

***** SETUP FOR LINKS.

PARAMETER MAXLKS=100
REAL LKPROB
COMMON /LINKS/ LKID(MAXLKS), LKLNG(MAXLKS), LKLANS(MAXLKS),
1 LKNOD(MAXLKS), LKPOA(MAXLKS), LKLEAD(MAXLKS), LKLAST(MAXLKS),
2 LKHD(MAXLKS), LKPROB(MAXLKS), LKDEST(MAXLKS), NUMLKS,
3 LKTD(MAXLKS), LKDEL(MAXLKS), LKDFLQ(MAXLKS), LKUL(MAXLKS),
4 LKV0D(MAXLKS), LKSTOP(MAXLKS), LKMAXQ(MAXLKS)

***** SETUP FOR VEHICLES.

PARAMETER MAXVEH=2000
INTEGER VLANE, VLKTIM, VLAD, VFLOW, VTURN, VTAG, VACC
COMMON /VEH/ VDSP(MAXVEH), VADP(MAXVEH), VAC(MAXVEH),
1 VPOS(MAXVEH), VLFAD(MAXVEH), VFLOW(MAXVEH), VTURN(MAXVEH),
2 VLKTIM(MAXVEH), VLANE(MAXVEH), VTAG(MAXVEH), VHOLD(MAXVEH),
3 NUMVEH

***** GENERAL COMMON VARIABLES.

INTEGER CLOCK, FINISH, WARMUP, DELT
COMMON CLOCK, ISEED, FINISH, WARMUP, DELT

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* DEFINITION OF VARIABLES USED IN THE PROGRAM. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
* AMBERT - AMBER TIME IN SECONDS.
* CLOCK - MASTER SIMULATION CLOCK.
* DELT - DELTA TIME, THE TIME STEP.
* FINISH - THE ABSOLUTE STOPPING TIME.
* ISEED - THE INITIAL RANDOM NUMBER SEED.
* ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.
* KTH - INDEX USED TO POINT TO SPECIFIC LINK. *
LKDEL(KTH,1-2) - CUMULATIVE DELAY ON LINK BY LANES.
LKDEL(KTH,1-3) - CUMULATIVE DELAY ON LINK BY DESTINATION.
LKDEST(KTH,1-3) - LINK DESTINATIONS: LEFT, STR RIGHT.
LKHOLD(KTH,1-2) - INDEX TO VEHICLE IN HOLDING AREA.
LKID(KTH,1-2) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).
LKLAN(KTH) - NUMBER OF LANES.
LKLAST(KTH,1-2) - INDEX TO LAST VEHICLE.
LKLEAD(KTH,1-2) - INDEX TO LEADER.
LKLEN(KTH) - LINK LENGTH (IN FEET).
LKMXX(KTH,1-2) - MAXIMUM QUEUE LENGTH, BY LANES.
LKNOD(KTH) - INDEX TO Node AT HEAD OF LINK.
LKOPOS(KTH,1-2) - INDEX TO NODE AT HEAD OF LINK.
LKPOR(KTH,1-2) - CUMULATIVE PROB OF LEFT & STR. MOVE.
LSTOP(KTH,1-2) TOTAL NUMBER OF STOPS ON LINK BY LANES.
LTA(KTH,1-2) - LINK TAG INDICATOR:
 0 - OK, 1 - MUST TAG A VEHICLE TO STOP.
LKD(KTH,1-3) - VEHICLE COUNT BY DESTINATION.
LKVD(KTH,1-3) - VEHICLE COUNT BY DESTINATION.
MAXLKS - MAXIMUM POSSIBLE NUMBER OF LANES.
MAXNO - MAXIMUM POSSIBLE NUMBER OF NODES.
MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.
MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.
NOCLK(NTH) - TIME OF NEXT GENERATION FROM NODE.
NOINO(NTH) - SYMBOLIC NODE IDENTIFIER.
NOINP(NTH,1-4) - INDEX TO LINKS INPUTTING TO THIS NODE.
NOINX(NTH,1-4) - SYMBOLIC IDENTIFIERS OF ADJ. NODES.
NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE.
NOPARS(NTH,1-2) - PARAMETERS FOR GENERATE NODES.
NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.
 NOTYPE(NTH) - NODE TYPE:
 0 - BOTH GENERATE AND TERMINATE,
 1 - GENERATE ONLY,
 2 - TERMINATE ONLY,
 4 - TWO WAY STOP,
 5 - FIXED TIMF CONTROLLED.
NTH - INDEX USED TO POINT TO SPECIFIC NODE.
NUMLKS - NUMBER OF LANES IN MODFL.
NUMNOS - NUMBER OF NODES.
NUMSIG - NUMBER OF SIGNALS.
NUMVEH - NUMBER OF VEHICLES IN NETWORK.
SIGCLK(NTH,1-2) - TIME OF NEXT STATE CHANGE TO SIGNAL.
SIGCYC(NTH) - CYCLE LENGTH OF ITH SIGNAL.
SIGDR(NTH,1-2) - LENGTH OF GREEN SIGNAL.
SINGP(NTH,1-4) - LINKS INPUTTING TO THE ITH SIGNAL.
SIGOFF(NTH,1-2) - OFFSET TIME.
SIGRED(NTH,1-2) - RED TIME FOR ITH SIGNAL.
SIGSTA(NTH,1-2) - SIGNAL STATE:
 0 - GREEN,
 1 - AMBER,
 2 - RED.
VACC(JTH) - ACCELERATION INDICATOR FOR LEFT TURNS.
VSP(JTH) - VEHICLE'S ACTUAL SPEED.
VDP(SJTH) - VEHICLE'S DESIRED SPEED.
VFOLLOW(JTH) - VEHICLE FOLLOWING THE JTH VEHICLE.
VHOLD(JTH) - NUMBER OF FEET MOV'D BEFORE HOLDING.
VLAVE(JTH) - LANE DESIGN.
VLEAD(JTH) - VEHICLE LEADING THE JTH VEHICLE.
VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK.
VLKTM(JTH) CLOCK TIME VEHICLE SHOULD LEAVE LINK.
VPDS(JTH) - POSITION OF JTH VEHICLE FROM END OF LINK.
VTAG(JTH) - STOPPING TAG INDICATOR, 1 - STOP.
VTURN(JTH) - TURN INDICATOR:
  1 - LFFT,
  2 - STRAIGHT,
  3 - RIGHT.
WARMUP = THF AMOUNT OF TIME FOR WARMUP.

END

THIS IS THE MAIN PROGRAM FOR THE CONTINUOUS MODEL.
CLOCK IN SECONDS.

INCLUDE DEFINS

***** INITIALIZATION
NTH = 0
DELT = 1
CLOCK = -DELT

***** RUN PARAMETERS
READ(5,1) IRUN,INET,AMBERT,ISEED,Fnish,WARMUP
1 FORMAT( )
MODEL = 'CONT'
WRITE(21) MODEL,IRUN,INET,ISEED
WRITE(22) MODEL,IRUN,INET,ISEED
PUNCH 6,IRUN,INET,ISEED
6 FORMAT('CONT MODEL IRUN = ',14,1,14,1,12)
WARMUP = WARMUP * 60
FINISH = FINISH * 60 + WARMUP
WRITE(6,2) IRUN,INET,ISEED,Fnish,WARMUP
2 FORMAT('1*************** CONTINUOUS MODEL ***************',//
1 ' RUN NUMBER ',15,' NETWORK ',15,' ISEED ',19
2 '// Finish = ',19,' SECONDS. WARMUP = ',19,' SECONDS.',//)

***** CALL INPUT SUBROUTINES.
CALL EXTERN(NTH)
CALL INTERN(NTH)
CALL INPLKS

***** CALL SETUP TO FINISH DATA MANIPULATION.
CALL SETUP
CALL DUMP(0,*DATA C++HECK *,n)

***** THE SIMULATION.

1000 CLOCK = CLOCK + DFLT
IF(MOD(CLOCK*90).EQ.0) CALL STATA(2)
WRITE(22) CLOCK,NUMVEH
IF(CLOCK .GE. WARMUP) GO TO 2000
CALL SIGCHK
CALL VEGEN
CALL LKMOVE
CALL HOLDOU
GO TO 1000

***** WARMUP OVER, CLEAR STATISTICS.

2000 CALL STATA(1)
CALL STATA(1)
GO TO 2200

2100 CLOCK = CLOCK + DFLT
IF(MOD(CLOCK*90).EQ.0) CALL STATA(2)
WRITE(22) CLOCK,NUMVEH
IF(CLOCK .GE. FINISH) GO TO 3000
CALL SIGCHK
CALL VEGEN
CALL LKMOVE
CALL HOLDOU
GO TO 2100

***** SIMULATION OVER, PRINT STATISTICS.

3000 CALL STATA(2)
ENDFILE 21
ENDFILE 22
STOP
END

SUBROUTINE DUMP(M,N1,N2,K)

***** THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
IN CASE OF AN ERROR. THE VALUE OF K DETERMINES
THE TYPE OF DUMP:

0 = SFT UP CHKCK, CALL RETURN
   AFTER PRINTOUT ON NODES, FTC
1 = ERROR ON INPUT,
>1 = ERROR WHILE RUNNING - LINE NUMBER.

INCLUDE DEFINS

***** GENERAL INFORMATION DUMP.

WRITE(6,1) K
1 FORMAT(/' ERROR DUMP OUTPUT, K =*15)
100 WRITE(6,3) M,N1,NP
3 FORMAT(* THING *15,** NOW BEING PROCESSED.*/* MESSAGE -'
   I,1X,12A6)
C  ***** NODE OUTPUT.
C
1010 FORMAT(///, 'NUMBER = ',I2, 'INDEX ID NEXT NODE ID', 'TYPE INDEX LINKS IN LINK OUT SIGNAL', 'CLOCK PARAMETERS')
1010 I = 1, NUMNOS
1010 WRITE(6, 1011) I, NOIDNO(I), (NOXT(I, K), K = 1, 4), NTYPE(I), (NOPARNS(I, K), K = 1, 2)
1011 FORMAT(14, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X)
1011 WRITE(6, 1012) I, NOIDNO(I), (NOXT(I, K), K = 1, 4), NTYPE(I), (NOPARNS(I, K), K = 1, 2)
1012 FORMAT(14, 5X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X)
C  ***** SIGNAL OUTPUT
C
1020 FORMAT(///, 'SIGNAL OUTPUT MAX = ',I2, 'NUMBER = ',I2, 'AMBER TIME = ',I2, '23X', 'MAJOR*', 'MINOR', '***', 'INDEX CYCLE', '26X', 'OFFSET GREEN RED STATE CLOCK', 'CLOCK PARAMETERS')
1020 I = 1, NUMSIG
1020 WRITE(6, 1021) I, SIGCYC(I), (SIGOFF(I, K), SIGGRN(I, K), SIGRED(I, K), SIGSTG(I, K), SIGCLK(I, K), K = 1, 2)
1021 FORMAT(14, 5X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X)
C  ***** LINK OUTPUT.
C
1030 FORMAT(///, 'LINK OUTPUT MAX = ',I3, 'NUMBER = ',I3, 'INDEX ID NO. LENGTH LANES NODE LK OPOS LK*', 'DESTINATIONS PROBABILITIES LEAD LEAD LAST LAST TAG', 'HOLD AREA')
1030 I = 1, NUMLKS
1030 WRITE(6, 1031) I, LKID(I, 1), LKIN(I, 2), LKLNG(I, 1), LKLANS(I), LKNOA(I, 1), LKPOS(I, 1), (LKDEST(I, K), K = 1, 4), (LKPROR(I, K), K = 1, 2)
1031 FORMAT(14, 4X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X)
1031 WRITE(6, 1032) I, LKID(I, 1), LKIN(I, 2), LKLNG(I, 1), LKLANS(I), LKNOA(I, 1), LKPOS(I, 1), (LKDEST(I, K), K = 1, 4), (LKPROR(I, K), K = 1, 2)
1032 FORMAT(14, 4X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X, I2, 3X, 6X)
1032 IF(K.EQ.0) RETURN
1032 IF(K.EQ.1) STOP
C  ***** VEHICLE OUTPUT.
C
1050 FORMAT(///, 'VEHICLE OUTPUT MAX = ',I4, 'NUMBER = ',I4, 'INDEX DSP ASP LINK LANE TURN POSITION ACC TAG', 'LEAD FOLLOW HOLD')
1050 I = 1, MAXVEH
1050 WRITE(6, 1051) I, MAXVEH, VDSP(I), .LE. 0)
1051 Go To 105
WRITE(6,1051) VN, VSP(I), VASP(I), VLK(I), VLANE(I), VTURN(I),
1 VPOS(I), VACC(I), VTAG(I), VLEAN(I), VFNLOW(I), VHOLD(I)
1051 FORMAT (14,5X,F3.0, 3X, T3, 6X, I1, 6X, I1, 5X, F5.0, 6X, I1, 5X, T1,
1 4X, 14, 4X, 14, 4X, F3.0)
105 CONTINUE
STOP
END
SUBROUTINE EXTERN(NTH)
C
C THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C NODES ONLY. CONTROL IS RETURNED TO THE CALLING
C PROGRAM AFTER A BLANK CARD IS FOUND
C
INCLUDE DEFINS
INTEGER ALPHA
1000 NTH=NTH+1
READ (5,101) ALPHA, NOIDNO(NTH), NOTYPE(NTH), NONXT(NTH+1),
1 NOPARS(NTH+1), NOPARS(NTH+2)
IF(ALPHA .EQ. 1H ) GO TO 2000
GO TO 1000
2000 NTH=NTH-1
RETURN
101 FORMAT (A1, I3, I2, I3, 2F5.0)
END
SUBROUTINE GAPCHK(KTH, KOUT, III)
C
C THIS SUBROUTINE FINDS THE SIZE OF THE GAP ON THE
C KTH LINK AND TESTS FOR LEFT HAND TURNS. KOUT EQUALS
C 0 IF BIG ENOUGH AND 1 IF GAP TOO SMALL.
C
INCLUDE DEFINS
DIMENSION PROR(20)
DATA PROB / -1., -1., 02, -10, -22, -36, -50, -61, -74,
1 81, .94, .90, .92, .95, .96, .97, .98, .99, .995, .9995 /
C
C IF SIGNAL RED, DO NOT BOTHER WITH GAP CHECKING.
C
IF(SIGSTA(III,II) .EQ. 2) GO TO 2000
C
C BRANCH ON NUMBER OF LANES.
C
IF(LKLANS(KTH) .EQ. 2) GO TO 1300
C
C SINGLE LANE APPROACH.
C
JTH = LKLEAD(KTH+1)
1252 IF(JTH .EQ. 0) GO TO 2000
IF(VPOS(JTH) .LE. LKLNK(KTH) + 20) GO TO 1260
JTH = VFOLLOW(JTH)
GO TO 1252
1260 IF(VASP(JTH) .LE. 4.0) GO TO 2000
IF(VTURN(JTH) .EQ. 1) GO TO 2000
IF(VTAG(JTH) .EQ. 1) GO TO 2000
GAP = (LKLNK(KTH) + 20-VPOS(JTH))/VASP(JTH)
GO TO 1500
C
C TWO LANE APPROACH; FIND THE SHORTEST GAP.
GAP2 = (LKLNG(KTH) + 20 - VPOS(JTH))/VASP(JTH)

GAP = MIN(GAP1, GAP2)

TEST THE ACCEPTANCE DISTRIBUTION.

IF(GAP .GE. 20.0) GO TO 2000
IF(GAP .LE. 2) GO TO 1750
INDEX = GAP * 0.5
IF(RNKISEED) .LE. PROB(INDEX)) GO To 2000
C REJECT.

KOUT = 1
RETURN
C ACCEPT.

KOUT = 0
RETURN
END

SUBROUTINE HOLDIN(JTH, KTHOLD, KTHNEW, INDEX, DIST)

REMOVES VEHICLE FROM OLD LINK AND PLACES IT ON NEW.

IF(KTHOLD .EQ. 0) GO TO 500

IF(KTHOLD .EQ. 0) GO TO 500

REMOVE FROM CHAIN ON OLD LINK.

ILANE = VLANE(JTH)
JTHNXT = VFOLLOW(JTH)
LKLEAD(KTHOLD, ILANE) = JTHNXT
ITURN = VTURN(JTH)
DELAY = CLOCK - VTKTIM(JTH)
WRITE(21) CLOCK, KTHOLD, JTH, DELAY
LKDEL(KTHOLD, ILANE) = LKDEL(KTHOLD, ILANE) + DELAY
LKDEL(KTHOLD,ITURN) = LKDEL(KTHOLD,ITURN) + DELAY
LKVL(KTHOLD,ILANE) = LKVL(KTHOLD,ILANE) + 1
LKVD(KTHOLD,ITURN) = LKVD(KTHOLD,ITURN) + 1
IF(JTHNXT .EQ. 0) GO TO 100
VLEAD(JTHNXT) = 0
VFOLLOW(JTH) = 0
GO TO 500
100 LKLAST(KTHOLD,ILANE) = 0
C
C ***** PLACE IN HOLDING AREA
C
500 LKHOLD(KTHNEW,INDEX) = JTH
VHOLD(JTH) = DIST
C**** WRITE(6,1) JTH,KTHNEW,INDEX,DIST
1 FORMAT(* HOLDING VEH 'I4' FROM LINK 'I3' TO LINK 'I3',
1 12, DIST 'F3.0')
RETURN
END
SUBROUTINE HOLDOU
C
C ***** THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING
C AREA AND MOVES THEM ONTO THE LINK
C
INCLUDE DEFINES
DIMENSION LKFAC(T,3,3)
DATA LKFAC / 188,232, 30,60, 145,145, 356,397, 30,60, 273,273,
1 489,568, 30,60, 412,412 /
DEFINE LFACT(LANE,TURN,ISPEED) = LKFAC(LANE,TURN,ISPEED)
C**** WRITE(6,1)
1 FORMAT(' HOLDOU')
DO 3000 KTH=1,NUMLKS
LANE = 1
C
C ***** BRANCH ON NUMBER OF LANES
IF(LKLANS(KTH) .EQ. 2) GO TO 7000
C
C SINGLE LANE. LOOK AT HOLDING AREA. IF EMPTY SKIP.
LANE = 1
INDEX = 1
JTH = LKHOLD(KTH,1)
IF(JTH .EQ. 0) GO TO 3000
C
C IF NEW VEHICLE TO THIS LINK, ASSIGN PARAMETERS.
IF(VLK(JTH) .EQ. KTH) GO TO 1000
VLK(JTH) = KTH
VLANE(JTH) = 1
X = RNI(1,ISeed)
DO 1050 I=1,2
1050 CONTINUE
VTURN(JTH) = 3
GO TO 1075
1075 VTURN(JTH) = 1
1081 VLKTIME(JTH) = CLOCK + (LKLNK(KTH) + VHOLD(JTH))
1 + LFACT(1,VTURN(JTH),1) + VDSP(JTH)) / VDSP(JTH)
GO TO 1100
1082 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH))
1 + LFACT(1*VTIN(JTH)*2) - VDSP(JTH)) / VDSP(JTH)
GO TO 1100
1083 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH))
1 + LFACT(1*VTIN(JTH)*3) - VDSP(JTH)) / VDSP(JTH)
1100 ZA = ZACCEL(VASP(JTH),VDSP(JTH))

C

JTHCK = LKLAST(KTH)1)
IF(JTHCK .EQ. 0) GO TO 2900
ZS = ZSPACE(0,0+VASP(JTH)*VPOS(JTH)+VHOLD(JTH)*VASP(JTHCK))
Z = MIN(ZA*ZS) - VHOLD(JTH)
IF(Z .LT. 0) GO TO 2950
GO TO 2975

C

***** TWO LANE LINK PROCESSING. LOOK AT HOLDING AREAS.

2000 INDEX = 1
JTH = LKHOLD(KTH)1)
IF(JTH .NE. 0) GO TO 2100
2001 INDEX = 2
JTH = LKHOLD(KTH)2)
IF(JTH .EQ. 0) GO TO 3000
2100 ZA = ZACCEL(VASP(JTH),VDSP(JTH))

C**** WRITE(6,5) ZA, VASP(JTH), VDSP(JTH)
5 FORMAT()

C

***** FOUND A VEHICLE IN THE HOLDING AREA. IF VEHICLE
NEW TO THIS LINK, ASSIGN PARAMETERS.

IF(VLK(JTH) .EQ. KTH) GO TO 2250
VLK(JTH) = KTH
X = RN1(1SEED)
IF(X .GT. LKPROB(KTH)1)) GO TO 2140
VTURN(JTH) = 1
VLANE(JTH) = 1
GO TO 2149
2140 IF(X .GT. LKPROB(KTH)2)) GO TO 2145
IF(VTURN(JTH) .EQ. 2) GO TO 2149
VTURN(JTH) = 2
VLANE(JTH) = RN1(1SEED) * LKLANS(KTH) + 1
GO TO 2149
2145 VTURN(JTH) = 3
VLANE(JTH) = LKLANS(KTH)
2149 IF(IFIX(VDSP(JTH)) = 54) 2181,2182,2183
2181 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH))
1 + LFACT(2*VTURN(JTH)*1) - VDSP(JTH)) / VDSP(JTH)
GO TO 2250
2182 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH))
1 + LFACT(2*VTURN(JTH)*2) - VDSP(JTH)) / VDSP(JTH)
GO TO 2250
2183 VLKTIM(JTH) = CLOCK + (LKLNG(KTH) + VHOLD(JTH))
1 + LFACT(2*VTURN(JTH)*3) - VDSP(JTH)) / VDSP(JTH)

C

***** FIND LEADER IN DESIRED LANE.
```
2250 LANE = VLNE(JTH)
   JTHCHK = LKLAST(KTH+LANE)
   IF(JTHCHK = EQ. 0) GO TO 2900
   ZS = ZSPACE(0,0,VASP(JTH),VP0S(JTHCHK)+VHOLD(JTH),VASP(JTHCHK))
C**** WRITE(6+5) JTH,JTHCHK,ZS,VP0S(JTHCHK),VASP(JTHCHK)
   Z = MIN(ZA,ZS) - VHOLD(JTH)
   IF(Z .LT. ZA) VASP(JTH) = ZS
   IF(Z .LT. 0) GO TO 2950
   GO TO 2975
C
   ***** ADVANCE ZA, NO OTHER VEHICLE ON LINK.
C
2900 ZA = ZA - VHOLD(JTH)
   IF(ZA .LE. 0) GO TO 2950
   LHOLD(KTH,INDEX) = 0
   VP0S(JTH) = ZA
   VASP(JTH) = 2*(ZA+VHOLD(JTH)) - VASP(JTH)
   LKLEAD(KTH,LANE) = JTH
   LKLAST(KTH,LANE) = JTH
   VHOLD(JTH) = 0
C**** WRITE(6+2) JTH,KTH,VP0S(JTH),VASP(JTH)
   2 FORMAT(1 VEH 'I4,' ON LINK 'I3,' MOVES TO POSITION 'F3.0,'
   1 'SPEED = 'F4.1)
   GO TO 2999
C
   ***** DONT MOVE FROM HOLDING AREA.
C
2950 VASP(JTH) = 0
   VHOLD(JTH) = 0
   VP0S(JTH) = 0
C**** WRITE(6+3) JTH,KTH
   3 FORMAT(1 VEH 'I4,' STAYS IN HOLDING AREA ON LINK 'I3)
   GO TO 2999
C
   ***** ADVANCE TO 7.
C
2975 LHOLD(KTH,INDEX) = 0
   LKLAST(KTH,LANE) = JTH
   VP0S(JTH) = Z
   VASP(JTH) = 2*(Z+VHOLD(JTH)) - VASP(JTH)
   VLEAD(JTH) = JTHCHK
   VHOLD(JTH) = 0
   VFOLOW(JTHCHK) = JTH
C**** WRITE(6+4) JTH,KTH,VP0S(JTH),JTHCHK
   4 FORMAT(1 VEH 'I4,' ON LINK 'I3,' MOVES TO POSITION 'F3.0,'
   1 'BEHIND VEH 'I4)
2999 IF(INDEX .NE. LKLANS(KTH)) GO TO 2001
3000 CONTINUE
RETURN
END
FUNCTION IBHOI
D(KTH)
C
   THIS FUNCTION SEARCHES THE HOLDING AREA OF THE
   KTH LINK. IF THERE IS AN EMPTY SPACE IN THE
   HOLDING AREA, THE LANE INDEX IS RETURNED AS
   THE VALUE OF THE FUNCTION. IF THE AREA IS FULL,
   ZERO IS THE RETURNED VALUE OF THE FUNCTION.
```
INCLUDE DEFIN

***** BRANCH BASED ON THE NUMBER OF LANES.
IBHOLD = 0
IF(LKLANS(KTH) .EQ. 2) GO TO 1000

***** SINGLE LANE, IF HOLDING AREA FULL THEN RETURN.
IF(LKhold(KTH,1) .NE. 0) RETURN
IBHOLD = 1
RETURN

***** TWO LANES - MUST CHECK BOTH HOLDING AREAS.
1000 IF(LKhold(KTH,1) .NE. 0) GO TO 1010
IBHOLD = 1
RETURN
1010 IF(LKhold(KTH,2) .NE. 0) RETURN
IBHOLD = 2
RETURN
END

SUBROUTINE INPLKS

***** THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
CALCULATES SOME CROSS REFERENCE INDICES.

INCLUDE DEFIN

***** THIS IS THE LOCAL DECLARATION.

INTEGER ALPHA
KTH=0
1000 KTH=KTH+1
READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
LKNOS(KTH),LKPROB(KTH,1),LKPROB(KTH,2),LKDST(KTH,K),K=1,3
101 FORMAT(11,2L3,14,1I4,3L1,13,2F3.3,313)

***** IF IT IS A BLANK CARD, GO TO 2000.
IF(ALPHA .EQ. 1H) GO TO 2000

***** FIND NODE NUMBER OF THE HEAD NODE.
DO 1010 NTH=1,NUMNOS
IF(NOIDNO(NTH) .EQ. LKID(KTH,P)) GO TO 1015
1010 CONTINUE
CALL DUMP(0,'INPLKS',*DO1010',1)
1015 LKNOD(KTH)=NTH

***** SET UP THE TURNING PROBABILITIES.
1060 LKPROB(KTH,2)=LKPROB(KTH,1)*LKPROB(KTH,2)
IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
GO TO 1000

***** PRELIMINARY CALCULATIONS.
117

2000 NUMLKS=KTH-1
    IF(NUMLKS .GE. MAXLKS) CALL DUMP(NUMLKS,'INPLKS','MAXLKS',1)

C    FIND LINK OPPOSING A LEFT TURN.
C
   DO 2500 KTH=1[NUMLKS
   IF(LKOPOS(KTH) .EQ. 0) GO TO 2500
   DO 2400 KTH2=1[NUMLKS
   IF(LKID(KTH2) .NE. LKID(KTH)) GO TO 2400
   IF(LKOPOS(KTH) .EQ. LKID(KTH)) GO TO 2450
2400 CONTINUE
   CALL DUMP(KTH,'INPLKS','LKOPOS',1)
2450 LKOPOS(KTH)=KTH2
2500 CONTINUE
   RETURN
   END

C    THIS SUBROUTINE READS DATA CARDS ON INTERIOR
C    NODES ONLY. CONTROL IS RETURNED TO THE CALLING
C    PROGRAM AFTER A BLANK CARD IS FOUND.
C
   INTEGER ALPHA,TEMP
   INCLUDE DEFINES
   ITH = 0
1000 NTH = NTH + 1
   READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,T),T=1,4)
   IF(ALPHA .EQ. 1H ) GO TO 3000

C    IF NONSIGNAIZED RETURN TO READING NEXT NODE.
C
   IF(NOTYPE(NTH) .LT. 4) GO TO 1000

C    SIGNALIZED INTERSECTIONS REQUIRF ADDITIONAL CARDS.
C
   ITH = ITH + 1
   NOSIG(NTH) = ITH
   READ(5,102) ALPHA,KDUM,SIGCYC(I'TH),SIGOFF(I'TH,1),SIGGRN(I'TH,1),
   1 SIGOFF(I'TH,2),SIGGRN(I'TH,2)

C    CALCULATE STATE CHANGE TIMES.
C
   DO 2100 I=1,2
   SIGRED(I'ITH,1) = SIGCYC(I'TH) - SIGGRN(I'TH,1) - AMBERT
   TEMP = SIGOFF(I'TH,1) + SIGGRN(I'TH,1)
   IF(TEMP .GT. SIGCYC(I'TH)) GO TO 2100
   TEMP = TEMP + AMBERT
   IF(TEMP .GT. SIGCYC(I'TH)) GO TO 2010
   SIGSTA(I'TH,1) = 2
   SIGCLK(I'TH,1) = SIGOFF(I'TH,1)
   GO TO 2100
2005 SIGSTA(I'TH,1) = 1
   SIGCLK(I'TH,1) = TFMP - SIGCYC(I'TH)
   GO TO 2100
2010 SIGSTA(I'TH,1) = 0
   SIGCLK(I'TH,1) = TFMP - SIGCYC(I'TH)
   GO TO 2100

C
2100 CONTINUE
   GO TO 1000
3000
   NUNNOS = NTH - 1
   IF (NUMNOS .GE. MAXNO) CALL DUMP(NUMNOS,'INTERN','MAXNO',1)
   NUMSIG = ITH
   IF (NUMSIG .GE. MAXSIG) CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
   RETURN
   C
   C FORMAT STATEMENTS
   C
   101 FORMAT(A1,I3,T2,I3,I1)
   102 FORMAT(A1,I3,I3)
END
SUBROUTINE LKMOVE
   C
   C THIS SUBROUTINE MOVES VEHICLES ON EACH LINK.
   C
   INCLUDE DEFINES
   C
   DIMENSION IARM(4,2),LKEND(3,2)
   DEFINE LSTOP(KTH,LANE) = LKSTOP(KTH,LANE)
   DATA IARM/4,1,2,3,2,3,4,1/
   DATA LKEND / 0,3,12,85,60,1/
   INTEGER VTP,VT
   C** *PITE(6,1)
   1 FORMAT(' LKMOVE')
   DO 1500 NTH = 1,KUNNOS
   DO 1450 ITHARM = 1,4
   KTH = NOINP(NTH,ITHARM)
   IF(KTH .EQ. 0) GO TO 1500
   ISW = 1
   ZRVELT = 100.
   LENGTH = LKLNG(KTH)
   C
   C AN EXTERNAL NODE COULD NOT BE BLOCKED
   C
   IF (NOTYPE(NTH) .LT. 3) GO TO 100
   ZB1 = 100.
   ZB2 = 100.
   C
   C ..... CHECK LEFT AND RIGHT APPROACHES.
   C
   DO 150 I=1,2
   INDEX = IARM(ITHARM,1)
   KTHCHK = NOINP(NTH,INDEX)
   ISW = LKLNS(KTHCHK)
   JTHCHK = LKLEAD(KTHCHK,1)
   IF (JTHCHK .EQ. 0) GO TO 100
   IF (VPOS(JTHCHK) .GT. LKLNG(KTHCHK) + 1) GO TO 155
   100 IF (ISW .EQ. 1) GO TO 150
   C
   C ..... CHECK NEXT LANE.
   C
   JTHCHK = LKLEAD(KTHCHK,2)
   IF (JTHCHK .EQ. 0) GO TO 150
   IF (VPOS(JTHCHK) .GT. LKLNG(KTHCHK) + 1) GO TO 155
   150 CONTINUE
GO TO 160

C 155 ZBI = 1.0

C 160 KTHCHK = LKPOs(KTH)
   JTHCHK = LKLEAD(KTHCHK + 1)
   IF(JTHCHK .EQ. 0) GO TO 170
   IF(VTURN(JTHCHK) .NE. 1) GO TO 170
   IF(VPOs(JTHCHK) .LE. LKLNG(KTHCHK) + 1 + ISW*6) GO TO 170
   ZB2 = 1.0

170 ZBDELT = MIN(ZBI, ZB2)

200 LANCES = LKLANS(KTH)
   DO 1400 LANE = 1, LANCES
      LKT = LKTAG(KTH, LANE)
      C FIND LEADER.
      JTH = LKLEAD(KTH, LANE)
      IF(JTH .EQ. 0) GO TO 1400
      JTHP = JTH
   300 X = VPOs(JTH)
      V = VASP(JTH)
      VD = VDSP(JTH)
      VT = VTURNJTH)
      ZA = 100.
      ZB = 100.
      ZD = 100.
      ZS = 100.
      ZT = 100.
      IF(JTH .NE. JTHP) GO TO 330
   310 IF(ZBDELT .GE. 100.) GO TO 400
      ZB = ZDECEL(DUM, ZBDELT + LKLN - X*V)
      GO TO 400
   330 IF(VTP .NE. 1) GO TO 340
      IF(XP .GT. LKLN + 20 + 6*ISW) GO TO 310
   340 ZS = ZSPACE(X*V,XP,VP)
   400 ZA = ZACCEL(V, VD)

C 165 NOW CHECK LINK OPPOSED

C 170 ZBDELT = MIN(ZBI, ZB2)

C 300 X = VPOs(JTH)
C 310 IF(ZBDELT .GE. 100.) GO TO 400
C 330 IF(VTP .NE. 1) GO TO 340
C 340 ZS = ZSPACE(X*V,XP,VP)
C 400 ZA = ZACCEL(V, VD)

C C FIND LEADER.
C C JTH = LKLEAD(KTH, LANE)
C C IF(JTH .EQ. 0) GO TO 1400
C C JTHP = JTH
C 300 X = VPOs(JTH)
C 310 IF(ZBDELT .GE. 100.) GO TO 400
C 330 IF(VTP .NE. 1) GO TO 340
C 340 ZS = ZSPACE(X*V,XP,VP)
C 400 ZA = ZACCEL(V, VD)

C       IS LINK TAGGED.
C 1 IF(LKT .EQ. 0) GO TO 430

C C CAN THIS VEHICLE STOP WITHIN DECELERATION LIMITS.
C DUM2 = 0.0
C DUM1 = ZDECEL(DUM, LKLN - X*12.0*V)
   IF(DUM2 .GE. 12.0*H. DUM1 .LE. 0) GO TO 450
   LKT = 0
   LKTAG(KTH, LANE) = 0
   VTAG(JTH) = 1
   ZD = DUM1
   GO TO 500
C
C IS VEHICLE TAGGED

430 IF(VTAG(JTH), EQ. 0) GO TO 450

C HAS SIGNAL CHANGED BACK TO GREEN.

ITH = NOSIG(NTH)
II = 2 - MOD(ITHARM, 2)
IF(SIGSTA(ITH, II), NE. 0) GO TO 435
VTAG(JTH) = 0
GO TO 450

435 ZD = ZDECEL(DUM2, LENGTH - 12, n, v)
GO TO 500

C TURNING RESTRICTION

450 IF(VT, EQ. 2) GO TO 500
WRITE(6, 7)
7 FORMAT(' TURNING RESTRICTION. ')
IF(X, LT. LENGTH - 175) GO TO 500
TP = LENGTH + 1
IF(VT, EQ. 1) TP = TP + 6 * ISW
ZT = ZTURN(X, V, TP, MZT, VTRNEW)

C MOVE LIMIT.

500 Z = MIN(ZA, ZB, ZD, ZS, ZT)
WRITE(6, 5) ZA, ZB, ZD, ZS, ZT
5 FORMAT(' ZA, ZB, ZD, ZS, ZT: ', 5F6.1)
XNEW = X + Z
WRITE(6, 9) XNEW, TP, VACC(JTH)
9 FORMAT(' LEFT TURN XNEW, TP, VACC(JTH): ', 2F6.1, I3)
IF(XNEW, LT. TP) GO TO 1000
IF(VACC(JTH), NE. 0) GO TO 610
IF(JTHP, NE. JTH, AND. VTP, EQ. 1) GO TO 900

C TEST GAP AND HOLDING AREA.

CALL GAPCHK(LKXOPC(KTH), KOUT, NOSIG(NTH) * 2 - MOD(ITHARM, 2))
WRITE(6, 8) KOUT
8 FORMAT(' GAPCHK OUT = ', ZI2)
IF(KOUT, EQ. 1) GO TO 900
IF(ISHOLD(LKDEST(KTH)), EQ. 0) GO TO 900

C START PROCESSING THROUGH LEFT TURN.

VACC(JTH) = 4
IF(MZT, EQ. 2) GO TO 970
GO TO 620

***** PAST TURNING POINT AND ACCELERATING.*****

610 IF(VACC(JTH) .NE. 1) VACC(JTH) = VACC(JTH) - 1
620 VNEW = V + 5 + (VACC(JTH) - 1)
IF(VNEW .LT. VD) VNEW = VD
ZA = 0.5 * (V + VNEW)
XNEW = X + ZA
DIST = XNEW - LENGTH - LKEND(ISW)
IF(DIST .LE. 0) GO TO 1000
VACC(JTH) = 0
KTHCHK = LKDEST(KTH+1)
INDEX = IBHOLD(KTHCHK)
GO TO 950

***** STRAIGHT TRAFFIC.*****

700 IF(XNEW .LT. LENGTH) GO TO 1000
KTHCHK = LKDEST(KTH+2)
IF(IBHOLD(KTHCHK) .NE. 0) GO TO 730
JTHCHK = LKHOLD(KTHCHK)
DUM1 = VASP(JTHCHK) - VHOLD(JTHCHK) + LENGTH + LKEND(ISW)
ZS = ZSPACE(X,V,DUM1,VASP(JTHCHK))
XNEW = MIN(XNEW,X + ZS)
IF(XNEW .LT. X) XNEW = X
GO TO 1000
730 DIST = XNEW - LENGTH - LKEND(ISW)
IF(DIST .LE. 0) GO TO 1000
INDEX = IBHOLD(KTHCHK)
GO TO 950

***** RIGHT TURN VEHICLES.*****

800 TP = LENGTH + 1
IF(XNEW .LT. TP) GO TO 1000
IF(IBHOLD(LKDEST(KTH+3)) .EQ. 1) GO TO 900
IF(M2T .EQ. 2) GO TO 970
DIST = XNEW - LENGTH - LKEND(ISW)
IF(DIST .LT. 0) GO TO 1000
KTHCHK = LKDEST(KTH+3)
INDEX = IBHOLD(KTHCHK)
GO TO 950

***** STOP VEHICLES.*****

900 XNEW = TP
GO TO 1000

***** VEHICLE IS GOING TO A NEW LINK.*****

950 JTHP = VFLOW(JTH)
CALL HOLDIN(JTH,KTH,KTHCHK,INDEX,XNEW-X,DIST)
IF(JTHP .NE. 0) GO TO 1400
JTH = JTHP
GO TO 300
**LEFT TURNING AND FLAG FROM ZTURN IS ON.**

970 IF(Z .NE. ZT) GO TO 1000

VASP(JTH) = VTRNE
VPOS(JTH) = XNEW
GO TO 1010

**APPROACHING EXTERNAL NODE.**

974 IF(XNEW .LT. LENGTH) GO TO 1000

CALL TERMIN(JTH,KTH,LANE,JTHP)

JTH = JTHP
IF(JTH .EQ. 0) GO TO 1400

GO TO 300

**MOVE DESIRED DISTANCE.**

1000 VPOS(JTH) = XNEW

VNEW = 2.0*(XNEW-X)-V

IF(VNEW .GE. 4.0 .OR. VASP(JTH) .LE. 4.0) GO TO 1005

LSTOP(VLK(JTH)*VLANE(JTH)) = LSTOP(VLK(JTH)*VLANE(JTH)) + 1

1005 VASP(JTH) = VNEW

IF(VASP(JTH) .LT. 0) VASP(JTH) = 0.0

**WRITE**

WRITE(6,2) JTH,KTH,LANE,VPOS(JTH),VASP(JTH),VT

2 FORMAT(' VEH ',I4,' ON LINK ',I3,I2,' POS ',F5.0,' ASP ',F5.1,' VTURN ',I2)

1010 JTHP = JTH

**WRITE**

WRITE(6,2) JTH,KTH,LANE,VPOS(JTH),VASP(JTH),VT

JTH = VFOLLOW(JTHP)

IF(JTH .EQ. 0) GO TO 1400

VP = VASP(JTHP)

XP = VPOS(JTHP)

VTP = VT

GO TO 300

**END OF LANE CHECK**

1400 CONTINUE

**END OF ARM CHECK**

1450 CONTINUE

**END OF NODE CHECK.**

1500 CONTINUE

RETURN

FUNCTION PN1(ISEED)

**THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED**

**RANDOM NUMBERS BETWEEN 0 AND 1**

ISEED = ISEED*1A533

IF(ISEED .LT. 0) ISEED = ISEED+34359738367+1

RN1 = ISEED*2.0**(-35)

RETURN
END

SUBROUTINE SETUP

...... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA FILES BEFORE THE SIMULATION BEGINS.

INCLUDE DEFINES

...... FOR EACH NODE

DO 2100 NTH=1,NUMNOS
IF(NOTYPE(NTH) GT 2) GO TO 1500
IF(NOTYPE(NTH) EQ 2) GO TO 2100

...... GENERATE NODES, SET TIME OF FIRST ARRIVAL.

NOPARS(NTH+1) = 3600. / NOPARS(NTH+1)
DO 1001 I=1,110
1001 A = RN1(ISEED)
NOSEED(NTH) = ISEED
NOCLK(NTH) = TBAGFN(NTH)
C**** WPITE(6rl) NTH»NOCLK(NTH)
1 FORMAT(• SETUP! NTH =',I2', NOCLK =',F7.2)

C FIND LINK TO PUT VEHICLES ON.

DO 1100 KTH=1,NUMLKS
IF(NONXT(NTH) .EQ. LKID(KTH) ) .AND. NOIDNO(NTH) EQ LKID(KTH+1)
GO TO 1110
1100 CONTINUE
CALL DUMP(NTH,'SETUP 'DO1100')
1110 NOOUTP(NTH) = KTH

...... IF GENERATE ONLY SKIP TO NEXT NODE.

IF(NOTYPE(NTH) EQ 1) GO TO 2100

...... FIND INDEX TO LINK POINTING AT THIS NODE.

1300 CONTINUE
DO 1350 KTH=1,NUMLKS
IF(NONXT(NTH) .EQ. LKID(KTH+1) ) .AND. NOIDNO(NTH) EQ LKID(KTH+2)
GO TO 1360
1350 CONTINUE
CALL DUMP(NTH,'SETUP 'DO1350')
1360 NOINP(NTH) = KTH
GO TO 2100

...... INTERNAL NODES

1500 CONTINUE
ITH = NOSIG(NTH)
IDTH = NOIDNO(NTH)

...... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS NODE.
DO 1600 KTH=1,NUMLKS
C C ***** CHECK FOR INPUT TO THIS NODE. C C IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600 C C ***** OK, NOW FIND OUT WHERE IT CAME FROM. C DO 1520 I=1,4
IF(LKID(KTH,I) .EQ. NONXT(NTH,I)) GO TO 1525
1520 CONTINUE
CALL DUMP(NTH,'SETUP','DO1520','D01520'))
1525 NOINP(NTH,I) = KTH
IF(ITH .NE. 0) STGINP(ITH,I) = KTH
1600 CONTINUE
C C C SETUP SIGNAL LINKS.
C ITH = NOSIG(NTH)
DO 2000 I=1,4
SIGINP(ITH,I) = NOINP(NTH,I)
2000 CONTINUE
2100 CONTINUE
C C C FOR EACH LINK FIND THE DESTINATIONS AND SET FLACKS.
C DO 3000 KTH=1,NUMLKS
LKTAG(KTH,1) = 1
LKTAG(KTH,2) = 1
DO 2500 I=1,3
IF(LKDEST(KTH,I) .EQ. 0) GO TO 2500
DO 2400 KTHCHK=1,NUMLKS
IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I)) GO TO 2400
IF(LKID(KTHCHK,1) .NE. LKID(KTH,2)) GO TO 2400
LKDEST(KTH,I) = KTHCHK
GO TO 2500
2400 CONTINUE
2500 CONTINUE
3000 CONTINUE
RETURN
END
SUBROUTINE SIGCHK
C C C ***** THIS SUBROUTINE CHECKS SIGNAL SETTINGS TO SEE IF C A CHANGE IN STATE IS NECESSARY.
C POSSIBLE STATES ARE:
C 0 - GREEN
C 1 - AMBER
C 2 - RED
C INCLUDE DEFINES
DO 2000 ITH=1,NUMSIG
DO 1000 I=1,2
C C C ***** CHECK CLOCK FIRST
C IF(SIGCLK(ITH,1) .GT. CLOCK) GO TO 1000
C BRANCH BASED ON OLD STATE.

IF(SIGSTA(I TH, I) = 1)

OLD STATE WAS GREEN.

100 SIGSTA(I TH, I) = 1
SICCLK(I TH, I) = SIGCLK(I TH, I) + AMBERT

C TAG INPUTS TO THIS SIGNAL.

KTH = SIGINP(I TH, I)
LKTAG(KTH, 1) = 1
IF(LKLANS(KTH) .GT. 1) LKTAG(KTH, 2) = 1
KTH = SIGINP(I TH, I + 2)
LKTAG(KTH, 1) = 1
IF(LKLANS(KTH) .GT. 1) LKTAG(KTH, 2) = 1

C SKIP OUT TO NEXT SIGNAL.

GO TO 2000

C OLD STATE WAS AMBER.

500 SIGSTA(I TH, I) = 2
SIGCLK(I TH, I) = SIGCLK(I TH, I) + SIGRED(I TH, I)
GO TO 1000

C OLD STATE WAS RED.

700 SIGSTA(I TH, I) = 0
SIGCLK(I TH, I) = SIGCLK(I TH, I) + SIGGRN(I TH, I)

C NOW CHECK LENGTH OF QUEUES FOR INPUTS WHICH TURNED.

KTH = SIGINP(I TH, I)

C CHECK FIRST LANE.

LKTAG(KTH, 1) = 0
JTH = LKLAST(KTH, 1)
IF(JTH .EQ. 0) GO TO 750
710 IF(VASP(JTH) .LE. 4.0) GO TO 740
JTH = VLEAD(JTH)
IF(JTH .EQ. 0) GO TO 750
GO TO 710
740 LQ = 1
741 JTH = VLEAD(JTH)
IF(JTH .EQ. 0) GO TO 745
LQ = LQ + 1
GO TO 741
745 IF(LQ .GT. LKMAXQ(KTH, 1)) LKMAXQ(KTH, 1) = LQ
750 IF(LKLANS(KTH) .EQ. 1) GO TO 800

C C

C C NOW CHECK SECOND LANE.

C C
LKTAG(KTH,2) = 0
JTH = LKLAST(KTH,2)
IF(JTH = EQ. 0) Go TO 800
760 IF(VASP(JTH) = LE. 4.0) Go TO 790
JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 800
Go TO 760
790 LQ = 1
791 JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 795
LQ = LQ + 1
Go TO 791
795 IF(LQ = GT. LKMAXQ(KTH,2)) LKMAXQ(KTH,2) = LQ

C ***** NOW CHECK NEXT INPUT LINK.

800 KTH = SIGINP(TTH,1+2)

C ***** CHECK FIRST LANE.

LKTAG(KTH,1) = 0
JTH = LKLAST(KTH,1)
IF(JTH = EQ. 0) Go TO 850
810 IF(VASP(JTH) = LE. 4.0) Go TO A40
JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 850
Go TO 810
840 LQ = 1
841 JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 845
LQ = LQ + 1
Go TO 841
845 IF(LQ = GT. LKMAXQ(KTH,1)) LKMAXQ(KTH,1) = LQ
850 IF(LKLANSA(KTH) = EQ. 1) Go TO 1000

C ***** CHECK SECOND LANE.

LKTAG(KTH,2) = 0
JTH = LKLAST(KTH,2)
IF(JTH = EQ. 0) Go TO 1000
860 IF(VASP(JTH) = LE. 4.0) Go TO A90
JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 1000
Go TO 860
890 LQ = 1
891 JTH = VLEAD(JTH)
IF(JTH = EQ. 0) Go TO 895
LQ = LQ + 1
Go TO 891
895 IF(LQ = GT. LKMAXQ(KTH,2)) LKMAXQ(KTH,2) = LQ

C ***** NEXT CHECK THE MINOR CLOCK.

C 1000 CONTINUE
2000 CONTINUE

C**** WRITE(6,1) CLOCK,((SIGCLK(I+1),SIGSTA(I+2)),I=1,NUMSIG)
1 FORMAT(16(I6,12))
RETURN
END
FUNCTION SPDDIS(ISEED)
DIMENSION ARY(4)
DATA ARY / -1., 0.07, 0.97, 1.0 /
X=RNI(ISEED)
DO 100 I=2,4
IF(ARY(I) .GE. X) GO TO 200
100 CONTINUE
200 SPDDIS = 1 * 18.0
RETURN
END

SUBROUTINE STAT(K)
C
C      .... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARUp
C      AND TlEN PRINTS THEM AFTER THE FINISH.
C
INCLUDE DEFINS
C
C      .... IF FINISH* GO TO 1000
C
IF(K .EQ. 2) GO TO 1000
DO 200 KTH = 1, NUMLK5
DO 100 I=1,2
LKMAXQ(KTH*1) = 0
LKSTOP(KTH*1) = 0
LKDEL(KTH*1) = 0
100 LKVL(KTH*1) = 0
DO 200 I=1,3
LKDELD(KTH*1) = 0
200 LKV D(KTH*1) = 0
RETURN
C
C      .... PRINT THE STATISTICS IN THIS SECTION.
C
1000 TOTIM = FINISH - WARMUP
WRITE(*,1)
1 FORMAT(1H10X,\VFHICLE COUNT,15X,\VFHICLE VOLUME,2X,1,1X,\LANCE 1 \LANE 2 \TOTAL 1 \LANE 2 \TOTAL,21X,2L9)
DO 1100 KTH = 1, NUMLK5
LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
LKVOL1 = LKVL(KTH,1) * 3600 / TOTIM
LKVOL2 = LKVL(KTH,2) * 3600 / TOTIM
LKVT = LKVOL1 + LKVOL2
LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
DO 1100 WRITE(6,2) KTH, LKLANS(KTH), LKVL(KTH,1), LKVL(KTH,2), LKVTOT, LKVOL1, LKVOL2, LKVLT, LKDELT, DELVL1, DELVL2, DELVLT
2 FORMAT(14X,2I9,2X,12X,2F9.2)
WRITE(*,3)
3 FORMAT(/14X,\VFHICLE STOPS,11X,\MAX QUEUES,14X,\TURNs,1X,\TURN DFLAY PER VEHICLE,\/)
2  LINK LANE 1 LANE 2 TOTAL LANE 1 LANE 2 LEFT',
3  STRAIGHT RIGHT LEFT STRAIGHT RIGHT LEFT',
4  STRAIGHT RIGHT,/
5  DO 1200 KTH=1,NUMLKS
6  LKSTOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
7  DELVL1 = LKDELT(KTH,1) / LKVD(KTH,1)
8  DELVL2 = LKDELT(KTH,2) / LKVD(KTH,2)
9  DELVL3 = LKDELT(KTH,3) / LKVD(KTH,3)
10  WRITE(6,4) KTH,LKSTOT(KTH,1),LKSTOP(KTH,2),LKSTOT,LKM,X0(KTH,1),
11    1 LKMAXQ(KTH,i) = (LKVD(KTH(i),i=1,3),LKDOL(KTH,1,i=1,3),
12    2 DELVL1,DELVL2,DELVL3
13  FORMAT(I4,3(I7,2X),6X,12,7X,12,2X,3(3X,15),2(4X,16),3X,16,
14    1 2X,3F9.2)
15  WRITE(6,5) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVL,
16    1 LKMAXQ(KTH,i) = 1 LKMAXQ(KTH,2),LKMAXQ(KTH,3),
17    1 2F9.2)
18  END
19
20  SUBROUTINE STATIA(I)
21    C
22    C PERIOD BY PERIOD OUTPUT GENERATOR LKSOUT IS AN ARRAY
23    C TELLING THE DESIRED OUTPUT LINKS.
INCLUDE DEFINS
DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
1   *DELY(4),NL1(4),NS1(4),NR1(4),DELY1(4)
DATA LKSOUT / I,5,0,0 /
PARAMETER N=2
GO TO (10,20),1
10 CONTINUE
C**** WRITE(6,4)
DO 15 II=1,N
L1(II)=0
DELY1(II)=0
LL1(II)=0
LS1(II)=0
15 LR1(II)=0
RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
20 CONTINUE
DO 30 II=1,N
MM = LKSOUT(II)
L1P = LKVL(MM,1) + LKVL(MM,2)
N1(II)=L1P - L1(II)
L1(II)=L1P
NL1(II)=LKVD(MM,1)-LL1(II)
LL1(II)=LKVD(MM,1)
NS1(II)=LKVD(MM,2)-LS1(II)
LS1(II)=LKVD(MM,2)
NR1(II)=LKVD(MM,3)-LR1(II)
LR1(II)=LKVD(MM,3)
IDELP = LKDEL(MM,1) + LKDEL(MM,2)
DELY1(II)=IDELP-DELY1(II)
DELY1(II)=IDELP
AVG1(II)=DELY1(II)/N1(II)
X1(II)=IDELP/FLOAT(L1P)
C**** WRITE(6,2) L1P,LKVD(MM,J),J=1,3)
C**** IDELP,X1(II)*N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
30 CONTINUE
PUNCH 3,CLOCK,NUMVEH,(N1(II),AVG1(II),X1(II)),II=1,N)
RETURN
1 FORMAT(/9H CLOCK = ,17,13H NUMVEH = #I7)
2 FORMAT(10X,16,315,18,F10.2,I1n,315,F8.2)
3 FORMAT(17,15,4(15,2F6,1))
4 FORMAT(1H1,12X,10=,CUMULATIVE,1AX,7HCURRENT,7X,11HLAST PERIOD,,
1 13X,25HVOL LF ST RT DFLAY,7X,3HMOE,7X,
2 25HVOL LF ST RT DEL/V)
END
FUNCTION TBAGEN(NTH)
C
C THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C AT THE NTH NODE. AN EXPONENTIAL TIME BETWEEN
C ARRIVALS IS ASSUMED.
C
INCLUDE DEFINS
TBAGEN = -ALOG(RN1(NOSEED(NTH))) * NOPARS(NTH,1)
RETURN
END
SUBROUTINE TERMIN(JTH,KTH,LANE,JTHCHK)
*** THIS SUBROUTINE TERMINATES VEHICLES THAT ARE LEAVING THE NETWORK.***

INCLUDE DEFINES

*** FIRST WORK ON LINK.***
JTHCHK = VFOLOW(JTH)
LKLEAD(KTH,LANE) = JTHCHK
IF(JTHCHK .NE. 0) VLFAD(JTHCHK) = 0
IF(JTHCHK .EQ. 0) LKLAST(KTH,LANE) = 0

*** NEXT ZERO DFSIRFD SPEED AND LINK NUMBER.***
VDSP(JTH) = 0
VLK(JTH) = 0
VTURN(JTH) = 0
VFOLOW(JTH) = 0
NUMVEH = NUMVFH - 1

WRITE(6,1) JTH,KTH,JTHCHK
1 FORMAT(' TERMINATE VEHICLE ',I4,' FROM LINK ',I3,' VNFXT ',I3)
RETURN
END

SUBROUTINE VEGEN

*** THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER VEHICLE.***

INCLUDE DEFINES

WRITE(6,1)
1 FORMAT(' VEGEN SUBROUTINE ')
DO 2000 NTH=1,NUMNOS
IF(NOTYPE(NTH) .GT. 1) GO TO 2000

*** CHECK TO SEE IF READY FOR ANOTHER VEHICLE***

1000 IF (CLOCK .LT. NOCLK(NTH)) GO TO 2000

*** FIND OUT IF HOLDING AREA IS AVAILABLE.***
KTH = NOOUTP(NTH)
INDEX = IARHOLD(KTH)
IF(INDEX .EQ. 0) GO TO 2000

*** HOLDING AREA READY, FIND AN AVAILABLE VEHICLE.***

1105 CONTINUE
DO 1110 JTH=1,MAXVEH
IF(VDSP(JTH) .LE. 0) GO TO 1121
1110 CONTINUE
CALL DUMP(0, 'VEGEN ', 'MAXVEH ',7)

*** NOW FILL IN HIS PARAMETERS.***
1121 VDSP(JTH) = SPDIS(ISFED)
VASP(JTH) = VDSP(JTH)
NUMVEH = NUMVEH + 1

C C PLACE VEHICLE IN HOLDING AREA C C***** WRITE(6,2) JTH,NTH,VDSP(JTH)
2 FORMAT(1VEH,14,AT NODE,12," VDSP ,F4.0)
CALL HOLDIN(JTH,O,KTH,INDEX,0,n)
C C C****** COMPUTE TIME OF NEXT ARRIVAL
C C NOCLK(NTH) = NOCLK(NTH) + TBA6F(NTH)
C C C READY FOR ANOTHER.
C
GO TO 1000
2000 CONTINUE
RETURN
END

FUNCTION ZACCEL(V,VMAX)
C C ACCELERATION RESTRICTION CALCULATION.
PARAMETER ABAR = 4
VNEW = V + ABAR
IF(VNEW .GT. VMAX) VNEW = VMAX
ZACCEL = 0.5*(V+VNEW)
RETURN
END

FUNCTION ZDECFL(DTEST,XGAP,V)
C C STOPPING RESTRICTION.
PARAMETER DBAR = 6
IF(XGAP .LE. 0) GO TO 15
DTEST = V**2 / (2.0*XGAP)
IF(DTEST .GE. DBAR) GO TO 30
15 ZDECEL = XGAP
RETURN
30 R = 0.0625*DTEST**2 - 0.25*DTEST*V + 0.5*DTEST*XGAP
IF(R .GT. 0) GO TO 35
R = 0
GO TO 40
35 R = SQRT(R)
40 ZDECEL = 0.5*V - 0.25*DTEST + R
RETURN
END

FUNCTION ZSPACE(X,V,XP,VP)
C C SPACING RESTRICTION CALCULATIONS.
PARAMETER DBAR = 6
PARAMETER PMIN = 22
C C CBRANCH ON CASE TYPE.
C
IF(V .LE. VP) GO TO 30

*** CASE I. ***

10 R = 9*DBAR**2/16 - DBAR*V/4 + .75*DBAR*VP + DBAR*(VP-XP-PMIN)/2
   IF(R .GT. 0) GO TO 15
   R = 0
   GO TO 20
15 R = SQRT(R)
20 ZSPACE = .5*(V+VP) - .75*DBAR + R
   RETURN

*** CASE II. ***

30 ZSPACE = (XP-XP-PMIN+V)/3.0
   RETURN
END

FUNCTION ZTURN(X,VrTP,VMZTRNFW)

*** THIS FUNCTION COMPUTES THE DESIRED MOVE THAT
WOULD BE REQUIRED TO SLOW TO THE TURNING SPEED
BY THE TURNING POINT. ***

PARAMETER DBAR = 6
PARAMETER VMAX = 12
PARAMETER ABAR = 3
MZT = 1
B = TP - X
IF(B .LE. 0) GO TO 40
   R = DBAR**2/16 + VMAX**2/0.25 - DBAR*V/0.25 + 0.5*DBAR*B
   IF(R .GT. 0) GO TO 20
   R = SQRT(R)
   GO TO 25
20 R = 0
25 ZTURN = 0.5*V - 0.25*DBAR + R
   IF(ZTURN .LE. B) GO TO 50

*** VEHICLE PASSED THE TURNING POINT. ***

VMXTP = SQRT(V**2 + 2*ABAR*B)
IF(VMXTP - 15.0) 40, 40
30 BT = 1.0 -(B+R)/(V+15.0)
   ZTURN = B + 15.0*BT + 1.5*BT**2
   VTRNEW = 15.0 + 3.0*BT
   MZT = 2
   RETURN
40 ZTURN = 100.0
50 RETURN
END
APPENDIX C

FORTRAN LISTING OF UB MODEL
DEFINS PROCEDURE

***** SETUP FOR NODES.

PARAMETER MAXNO=35
REAL NOPARS,NOLCK
COMMON /NODES/ NOTONO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
1 NOPARS(MAXNO,2),NODFOM(MAXNO),NOINP(MAXNO,4),NODOUTP(MAXNO),
2 NOSEED(MAXNO),NOFSTR(MAXNO),NOSIG(MAXNO),NOCLK(MAXNO),NMNOS

***** SETUP FOR SIGNALS.

PARAMETER MAXSIG=20
INTEGER AMBERT, SIGCLK, SIGCYC, SIGGRN, SIGINP, SIGOFF, SIGRED, SIGSTA
COMMON /SIG/ SIGCL_K(MAXSIG,2), SIGCYC(MAXSIG), SIGINP(MAXSIG,4),
1 SIGGRN(MAXSIG,2), SIGOFF(MAXSIG,2), SIGRED(MAXSIG,2),
2 SIGSTA(MAXSIG,2), NUMSIG, AMBERT

***** SETUP FOR LINKS.

PARAMETER MAXLKS=100
REAL LKPROB
COMMON /LINKS/ LKTD(MAXLKS,2), LKNG(MAXLKS), LKLAN(MAXLKS),
1 LKFSB(MAXLKS), LKNOH(MAXLKS), LKPOS(MAXLKS),
2 LKPROB(MAXLKS,2), LKNEST(MAXLKS,3), LKVD(MAXLKS,3), LKVL(MAXLKS,2),
3 LKDEL(MAXLKS,2), LKDELD(MAXLKS,2), LKMAXQ(MAXLKS,2),
4 LKSTTOP(MAXLKS,2), NUMLKS

***** SETUP FOR BLOCKS.

PARAMETER MAXBLK=5000
INTEGER BLKARY
COMMON /BLOCKS/ BLKARY(MAXBLK), LENGTH, NBLK

***** SETUP FOR VEHICLES.

PARAMETER MAXVEH=2000
INTEGER VDSP, VASP, VMOVE, VSTATE, VTURN, VHLHOLD, VLK, VLKTIM
COMMON /VEH/ VDSP(MAXVEH), VASP(MAXVEH), VMOVE(MAXVEH), VSTATE(MAXVEH),
1 VTURN(MAXVEH), VLK(MAXVEH), VHLHOLD(MAXVEH), VLK(MAXVEH), VLKTIM(MAXVEH),
2 VLKTIM(MAXVEH)

***** GENERAL COMMON VARIABLES.

INTEGER CLOCK, DELT, FINISH, WARMUP
COMMON CLOCK, ISEEN, DELT, FINISH, WARMUP

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* DEFINITION OF VARIABLES USED IN THE PROGRAM. * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* AMBERT - AMBER TIME.*
* BLKARY(IB) - ARRAY OF BLOCKS.*
CLOCK - MASTER SIMULATION CLOCK.

DELT - THE TIME STEP.

FINISH - RUNNING TIME (IN SECONDS).

ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.

KTH - INDEX USED TO POINT TO SPECIFIC LINK.

LENGTH - LENGTH OF A UNIT BLOCK (IN FEET).

LKDST(KTH) - LINK DESTINATIONS; LEFT, STR, RIGHT.

LKFS1B(KTH) - LINK'S FIRST BLOCK.

LKID(KTH) - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).

LKLNS(KTH) - NUMBER OF LANES.

LKLNG(KTH) - LINK LENGTH (IN BLOCKS).

LKNOD(KTH) - INDEX TO NODE AT HEAD OF LINK.

LKNTYP(KTH) - TYPE OF NODE APPROACHING:

0 - NO REASON TO STOP IF STRAIGHT.
1 - APPROACHING A SIGNAL.
2 - APPROACHING A STOP SIGN.

LKOPOS(KTH) - LINK APPROACHING A LEFT TURN.

LKPROB(KTH) - CUMULATIVE PROB OF LEFT & STR. MOVE.

MAXBLK - MAXIMUM POSSIBLE NUMBER OF BLOCKS.

MAXLKS - MAXIMUM POSSIBLE NUMBER OF LINKS.

MAXNO - MAXIMUM POSSIBLE NUMBER OF NODES.

MAXSIG - MAXIMUM POSSIBLE NUMBER OF SIGNALS.

MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.

MAXVEH - MAXIMUM POSSIBLE NUMBER OF VEHICLES.

NBLKS - NUMBER OF BLOCKS IN MODFL.

NOCLK(NTH) - TIME OF NEXT VEGEN.

NOFSTB(NTH) - NODES FIRST BLOCK.

NOGEOM(NTH) - GEOMETRY OF NODE:

EXTERNAL = NUMBER OF LANES.
INTERNAL = TYPE OF INTERSECTION:

1 - TWO BY TWO,
2 - FOUR BY TWO,
3 - FOUR BY FOUR.

NOIDNO(NTH) - SYMBOLIC NODE IDENTIFIER.

NOINP(NTH) - INDEX TO LINKS inputting TO THIS NODE.

NOOUTP(NTH) - INDEX TO LINKS TAKING FROM THIS NODE.

NOPARS(NTH) - PARAMETERS FOR GENERATE NODES.

NOSIG(NTH) - INDEX TO SIGNAL FOR NTH NODE.

NOTYPE(NTH) - NODE TYPE:

0 - BOTH GENERATE AND TERMINATE,
1 - GENERATE ONLY,
2 - TERMINATE ONLY,
5 - FIXED TIME CONTROLLED.

NTH - INDEX USED TO POINT TO SPECIFIC NODE.

NUMLKS - NUMBER OF LINKS IN MODFL.

NUMNOD - NUMBER OF NODES.

NUMSIG - NUMBER OF SIGNALS.

NUMVEH - NUMBER OF VEHICLES IN NETWORK.

SIGCLK(IITH) - TIME OF NEXT SIGNAL CHANGE.

SIGCyc(IITH) - CYCLE LENGTH OF ITH SIGNAL.

SIGGRN(IITH) - GREEN TIME.

SIGIND(IITH) - LINKS POINTING AT ITH SIGNAL.

SIGOFF(IITH) - SIGNAL OFFSET.

SIGRED(IITH) - RED TIME.

SIGSTA(IITH) - SIGNAL STATE:

0 - GREEN
1 - AMBER
2 - RED

* VASP(JTH) - VEHICLE'S ACTUAL SPEED
* VDSP(JTH) - VEHICLE'S DESIRED SPEED
* VHOLD(JTH) - NUMBER OF BLOCKS MOVED BEFORE HOLDING
* VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK
* VLAN(JTH) - LANE DESIRED
* VLK(JTH) - INDEX OF CURRENT OR MOST RECENT LINK
* VMOVE(JTH) - MOVE FLAG, n - NOT PROCESSED THIS PERIOD
* VSTATE(JTH) - STATE OF JTH VEHICLE:
  0 - STOPPED, MUST HESITATE
  1 - ACCELERATING
  2 - DECELERATING
  3 - CONSTANT SPEED
* VTURN(JTH) - TURN INDICATOR:
  1 - LEFT
  2 - STRAIGHT
  3 - RIGHT
* WARMUP - TIME TO LOAD THE NETWORK

END

***** THIS IS THE MAIN PROGRAM FOR THE UNIT BLOCK MODEL.
      CLOCK IN SECONDS

INCLUDE DEFINES.LIST

***** INITIALIZATION
NTH = 0
DELT = 1
CLOCK = -DELT
NBLKS = 0
LENGTH = 18

***** RUN PARAMETERS
READ(5,1) IRUN,INFT,AMBERT,ISEFD,FINISH,WARMUP
1 FORMAT( )
MODEL = ' UB'
WRITE(21) MODEL,IRUN,INFT,ISEFD
WRITE(22) MODEL,IRUN,INFT,ISEFD
PUNCH 6,IRUN,INFT,ISEFD
6 FORMAT('UB MODEL IRUN = '14,' INFT = '14,' ISEED = '12)
WARMUP = WARMUP * 60
FINISH = FINISH * 60 + WARMUP
WRITE(6*2) IRUN,INFT,ISEED,FINISH,WARMUP
2 FORMAT('1*************** UNIT BLOCK MODEL **********************',//
1 'RUN NUMBER '15'/' NETWORK '15' ISEED '19' // FINISH = '14' SECONDS. WARMUP = '14' SECONDS.' ISEED

..... CALL INPUT SUBROUTINES.
CALL EXTERN(NTH)
CALL INTERN(NTH)
CALL INPLKS

..... CALL SETUP TO FINISH DATA MANIPULATION.
CALL SETUP
CALL DUMP(0,'DATA CHECK',n)

..... THE SIMULATION.
1000 CLOCK = CLOCK + DFLT
IF(MOD(CLOCK,90),FQ,0) CALL STAT(2)
WRITE(22) CLOCK,NUMVEH
IF(CLOCK .GE. WARMUP) GO TO 2000
CALL VCLEAR
CALL SIGCHK
CALL VEGEN
CALL INTERL
CALL INTRAL
CALL HOLDOU
GO TO 1000

..... WARMUP OVER, CLEAR STATISTICS.
2000 CALL STAT(1)
CALL STAT(1)
GO TO 2200
2100 CLOCK = CLOCK + DFLT
IF(MOD(CLOCK,90),FQ,0) CALL STAT(2)
WRITE(22) CLOCK,NUMVEH
IF(CLOCK .GE. FINISH) GO TO 3000
2200 CALL VCLEAR
CALL SIGCHK
CALL VEGEN
CALL INTERL
CALL INTRAL
CALL HOLDOU
GO TO 2100

..... SIMULATION OVER, PRINT STATISTICS.
3000 CALL STAT(2)
ENDFILE 21
ENDFILE 22
STOP
END
SUBROUTINE ADVCHK(JTH,JTHDR,NUMPOS,NLIM)

..... THIS SUBROUTINE CALCULATES HOW FAR VEHICLES CAN
ADVANCE KNOWING SIZE OF GAP AND VEHICLES INVOLVED.
INCLUDE DEFINS

DIMENSION IGAPA(4,5)
DATA IGAPA/0,1,3,4,1,2,4,6,0,2,4,0,4,0,0,2,4,0,0,3/
DEFINE IGAPA(I,J)=IGAPA(I,J+1)
NUMS = VASP(JTH)
IF(VASP(JTH) .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1) NUMDS =
1
DO 1000 I=NUMDS,1,-1
NUM = 1
IF(NUMPOS-NUM .GE. IGAP(NUM,VASP(JTH))) RETURN
1000 CONTINUE
NUM = 0
1010 RETURN

END

SUBROUTINE DISTCH(JTH,KTH,POS,NTH,NUM,ITH,IARM)
C
C THIS SUBROUTINE CHECKS THE JTH VEHICLE WHICH IS
C IN POSITION POS OF THE KTH LINK APPROACHING THE
C NTH NODE. NUM2 IS THE NUMBER OF POSITIONS HE
C SHOULD MOVE CONSIDERING ONLY THE INTERSECTION.
C
INCLUDE DEFINS
C
DIMENSION ISQ(4,7,4)
INTEGER POS,EFFECT(4),TURNSP(10),ISQ(4,3),STOPSP(10)
DATA ISQ /3,4,2,1,4,2,1,3,2,1,3,4 /
DATA ISQ2 /9,10,11,12,13,14,15,16,15,11,7,3,16,2,8,4 /
1 8,7,6,5,4,3,2,1, 2,6,10,14,1,5,9,13/
DATA EFFECT /4,6,8,10 /
DATA TURNSP /1,2,2,3,3,4,4,4,4,4 /
DATA STOPSP /0,1,1,2,2,3,3,3,4,4 /
II = 2 - MOD(IARM,2)

C 
C ..... DETERMINE FASTEST SPEED DESIRED.
C NUM2 = VASP(JTH)
C IF(NUM2 .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1) NUM2 = NUM2 + 1
C
C ..... CHECK THE POSITION OF THE VEHICLE.
C IF(POS .GT. EFFECT(NUM2)) RETURN
C IF(NOTYPE(NTH) .LT. 2) RETURN
C IF(NOGEOM(NTH) .EQ. 3) GO TO 3000
C
C ..... TWO-BY-TWO INTERSECTION.
C
1050 IF(SIGSTA(IITH,II) .NE. 0) GO TO 9900
C IF(VTURN(JTH) .EQ. 2) GO TO 1065
C 1055 IF(NUM2 .GT. TURNSP(POS)) NUMP = TURNSP(POS)
C 1065 IF(POS-NUM2 .GT. 0) RETURN
C NUMIN = NUM2 - POS + 1
C IF(VTURN(JTH)-2) 2200*2100*2nn0
C
C ..... RIGHT TURNS.
C
2000 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
C IF(BLKARY(IB) .EQ. 0) GO TO 2100
C GO TO 9800
2010 IF(IBHOLD(LKDEST(KTH,3)) *EQ. 0) GO TO 9800
RETURN

2100 IB = ISQ(IARM,2) + NOFSTB(NTH) - 1
IF(BLKARY(IB) *EQ. 0) GO TO 2110
JTHLDR = ALKARY(IN)
IF(VL(JTHLDR) *NE. KTH) GO TO 9800
IF(VTURN(JTHLDR) *EQ. 2) GO TO 2106
GO TO 9800
2106 CALL ADVCHK(JTH, JTHLDR, POS+1, NUM)
IF(NUM1 LT. NUM2) NUM2 = NUM1
2110 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
IF(BLKARY(IB) *NE. 0) GO TO 9A00
IF(NUMIN LE. 2) RETURN
IF(IBHOLD(LKDEST(KTH,2)) *NE. n) RETURN
KTHLDR = LKDEST(KTH,2)
IB = LKFSTB(KTHLDR) + LKLNG(KTHLDR)
JTHLDR = ALKARY(IR)
CALL ADVCHK(JTH, JTHLDR, POS+1, NUM)
RETURN

2200 IF(IBHOLD(LKDEST(KTH,1)) *EQ. 0) GO TO 9800
IB = ISQ(IARM,2) + NOFSTB(NTH) - 1
IF(BLKARY(IB) *EQ. 0) GO TO 2210
JTHLDR = ALKARY(IN)
IF(VL(JTHLDR) *NE. KTH) GO TO 9800
IF(VTURN(JTHLDR) *EQ. 1) GO TO 9800
2210 IB = ISQ(IARM,1) + NOFSTB(NTH) - 1
IF(BLKARY(IB) *NE. 0) GO TO 9A00
RETURN

3000 IF(SIGSTA(IITH,11) = NE. 0) GO TO 9900
IF(VTURN(JTH) *EQ. 2) GO TO 3100
IF(NUM2 GT. TURNSP(POS)) NUM2 = TURNSP(POS)
3100 IF(POS-NUM2 GT. n) RETURN

Vehicles going into the intersection.

Right lane may turn right or go straight.

Right turns.

Four by four intersections.
C  *** STRAIGHT TRAFFIC IN THE RIGHT LANE.***

3200 NGAP = POS - 1
DO 3210 I=1,4
IB = ISO2(I,2,1ARN) + NOFSTB(I,TH) - 1
IF(BLKARY(IB) .NE. 0) GO TO 3220
3210 NGAP = NGAP + 1
RETURN

3220 JTHCHK = BLKARY(IR)
IF(VLK(JTHCHK) .NE. KTH) GO TO 9800
CALL ADVCHK(JTH,JTHCHK,NGAP,NUM1)
IF(NUM1 .LT. NUM2) NUM2 = NUM1
RETURN

C LEFT LANE, MAY TURN LEFT OR GO STRAIGHT*

3500 ISTOP = VTURN(JTH) + 2
NGAP = POS - 1
DO 3510 I=1,ISTOP
IB = ISO2(I,1,1ARN) + NOFSTB(I,TH) - 1
IF(BLKARY(IB) .NE. 0) GO TO 3520
3510 NGAP = NGAP + 1
RETURN

3520 JTHCHK = BLKARY(IR)
IF(VLK(JTHCHK) .NE. KTH) GO TO 9800
IF(VTURN(JTHCHK) .EQ. 1) GO TO 9800
CALL ADVCHK(JTH,JTHCHK,NGAP,NUM1)
IF(NUM1 .LT. NUM2) NUM2 = NUM1
RETURN

9800 NUM2 = POS - 1
RETURN

9900 IF(NUM2 .GT. STOPSP(POS)) NUMP = STOPSP(POS)
RETURN
END
SUBROUTINE DUMP(M,N1,N2,K)

C  *** THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
    IN CASE OF AN ERROR. THE VALUE OF K DETERMINES
    THE TYPE OF DUMP:
    0 - SFT UP CHCK, CALL RETURN
    AFTER PRINTOUT ON NODES, ETC.
    1 - ERROR ON INPUT,
    2 - ERROR WHILE RUNNING.

C INCLUDE DEFINES

C  *** GENERAL INFORMATION DUMP.***
WRITE(6,1) K
IF(K .GE. 2) GO TO 1000
WRITE(6,2) M,N1,N2
GO TO 1001
1000 WRITE(6,3) M,N1,N2
L = MAXNO
1001 WRITE(6,4) L,NUMNOS
DO 1010 I=1,NUMNOS
1010 WRITE(6,5) NODENO(I),NOTYPE(I),(NONXT(I,K),K=1,4),NOPARS(I,1),
1NOPARS(I,2),NOGEON(I),(NINP(I,K),K=1,4),NOOUTP(I),
2NOFSTB(I),NOSTG(I),NOCLK(I)
L = MAXLKS
WRITE(6,6) L,NUMLKS
DO 1020 I=1,NMLK
1020 WRITE(6,7) LKTU(I,1),LKD(I,2),LKLN(I),LKLS(I),LKFS(I),
   LKROPS(I),LKPRHS(I),LKSTATS(I),LKDETS(I,2),
   LKDETS(I,3)
WRITE(6,8) NBLKS,(BLKARY(I),I=1,NBLKS)
IF(K.EQ.0) RETURN
IF(K.EQ.1) STOP DUMP1
WRITE(6,9) CLOCK
DO 1090 J=1,MAVEH
IF(VDSP(J).EQ.0) GO TO 1090
IDSP=VDSP(J)
IASP=VASP(J)
MOVE=VMOVE(J)
STATE=VSTATE(J)
TURN=VTURN(J)
LK=VLK(J)
IHM=VHOLD(J)
IPLAN=VLAN(J)
WRITE(6,10) J,IDSP,IASP,MOVE,STATE,TURN,LK,IHM,IPLAN
1090 CONTINUE
STOP DUMP2
1 FORMAT(///ERROR DUMP OUTPUT, K='",15)
2 FORMAT(/// CODE='",15, MESSAGE='",2A6)
3 FORMAT(/// VEHICLE='",16, NOW BEING PROCESSED='", MESSAGE='",2A6)
4 FORMAT(/// NODE OUTPUT, MAX='",14, NUMBER='",15, ""\"2X"",2X, ""\"2X","",2X, ""\"2X","",1"",TYP*"",3X, ""\"NEXT NODE","",2X*PARAMETERS "",1O*3X,""\"LINKS IN","",4X,"",2*OUT*"",2X*FSTR*"",2X*STG*CLK*"",/)
5 FORMAT(2X,2X,12*X,3X,4*(I2,1X),2*(I2,F4.0),3*(3X),12*X,3*5,3*(2X,F5.3),12*X,3*5,3*(2X,F5.3),12*X,3*5)
6 FORMAT(/// LINK OUTPUT, MAX='",14, NUMBER='",15, ""\"3X","",3X, ""\"LENGTH LAKES","",3X,"",2*DESTINATIONS"",/)n
7 FORMAT(2X,2X,12*X,12*X,12*X,6*X,14*X,12*X,12*X,2*2X,F5.3,12*X,3*5)
8 FORMAT(/// BLOCK OUTPUT"",/(1X,25IS*))
9 FORMAT(/// CLOCK='",17, ""\"VEHICLES IN NETWORK"",/)
1P ASP MOVE STA TURN LK HOLD LAN*,/)
10 FORMAT(10I5)
END
SUBROUTINE EXTERN(NTH)
C C ..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C NODES ONLY. CONTROL IS RETURNED TO THE CALLING
C PROGRAM AFTER A BLANK CARD IS FOUND
C
C INCLUDE DEFINS
INTEGER ALPHA
1000 NTH=NTH+1
READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),NONXT(NTH*1),
   NOPARS(NTH*1),NOPARS(NTH*2),NOGEOM(NTH)
IF(ALPHA .EQ. 1H) GO TO 2000
GO TO 1000
2000 NTH=NTH-1
RETURN
**SUBROUTINE GAPCHK(KTH, KOUT, NTH, ITH, II)**

**C C ..... THIS SUBROUTINE FINDS THE SIZE OF THE GAP ON THE**
**KTH LINK.**

**C**

**INCLUDE DEFS**

**DIMENSION PROB(20)**

**DATA PROB / -.00, -.12, -.24, -.36, -.48, -.60, -.72, .84, .96, 1.08, 1.20, 1.32, 1.44, 1.56, 1.68, 1.80, 1.92, 2.04, 2.16, 2.28, 2.40, 2.52, 2.64, 2.76, 2.88, 3.00, 3.12, 3.24, 3.36, 3.48, 3.60, 3.72, 3.84, 3.96, 4.08, 4.20, 4.32, 4.44, 4.56, 4.68, 4.80, 4.92, 5.04, 5.16, 5.28, 5.40, 5.52, 5.64, 5.76, 5.88, 6.00, 6.12, 6.24, 6.36, 6.48, 6.60, 6.72, 6.84, 6.96, 7.08, 7.20, 7.32, 7.44, 7.56, 7.68, 7.80, 7.92, 8.04, 8.16, 8.28, 8.40, 8.52, 8.64, 8.76, 8.88, 9.00, 9.12, 9.24, 9.36, 9.48, 9.60, 9.72, 9.84, 9.96, 10.08, 10.20, 10.32, 10.44, 10.56, 10.68, 10.80, 10.92, 11.04, 11.16, 11.28, 11.40, 11.52, 11.64, 11.76, 11.88, 12.00, 12.12, 12.24, 12.36, 12.48, 12.60, 12.72, 12.84, 12.96, 13.08, 13.20, 13.32, 13.44, 13.56, 13.68, 13.80, 13.92, 14.04, 14.16, 14.28, 14.40, 14.52, 14.64, 14.76, 14.88, 15.00, 15.12, 15.24, 15.36, 15.48, 15.60, 15.72, 15.84, 15.96, 16.08, 16.20, 16.32, 16.44, 16.56, 16.68, 16.80, 16.92, 17.04, 17.16, 17.28, 17.40, 17.52, 17.64, 17.76, 17.88, 18.00, 18.12, 18.24, 18.36, 18.48, 18.60, 18.72, 18.84, 18.96, 19.08, 19.20, 19.32, 19.44, 19.56, 19.68, 19.80, 19.92, 20.00 /**KOUT = 0**
**IF(NOTYPE(NTH) .LT. 2) RETURN**
**IF(SIGSTA(ITH,II) .NE. 0) RETURN**
**1200 IF(LKLANS(KTH) .EQ. 2) GO TO 1300**

**C C ..... SINGLE LANE STREET.**

**IBSTRT = LKFSTB(KTH)**
**IBSTOP = IBSTRT + LKLNG(KTH) - 1**

**DO 1250 IR=IBSTRT,IBSTOP**
**IF(BLKARY(IB) .NE. 0) GO TO 1251**

**1250 CONTINUE**
**RETURN**

**1251 JTH = BLKARY(IB)**
**IF(VASP(JTH) .EQ. 0) RETURN**

**1260 GAP = (IR - IBSTRT) / FLOAT(VASP(JTH))**
**GO TO 1600**

**C C ..... TWO LANE APPROACH.**

**C C ..... CHECK THE FIRST LANE.**

**1300 GAP1 = 999.**
**IBSTRT = LKFSTB(KTH)**
**IBSTOP = IBSTRT + LKLNG(KTH) - 1**

**DO 1310 IR=IBSTRT,IBSTOP**
**IF(BLKARY(IB) .NE. 0) GO TO 1311**

**1310 CONTINUE**
**GO TO 1400**

**1311 JTH = BLKARY(IB)**
**IF(VASP(JTH) .EQ. 0) GO TO 1400**

**1320 GAP1 = (IR - IBSTRT) / FLOAT(VASP(JTH))**

**C C ..... CHECK THE SECOND LANE.**

**1400 GAP2 = 999.**
**IBSTRT = IBSTOP + 1**
**IBSTOP = IBSTRT + LKLNG(KTH) - 1**

**DO 1420 IR=IBSTRT,IBSTOP**
**IF(BLKARY(IB) .NE. 0) GO TO 1421**

**1420 CONTINUE**
**GO TO 1500**

**1421 JTH = BLKARY(IB)**
**IF(VASP(JTH) .EQ. 0) GO TO 1500**

**1430 GAP2 = (IR - IBSTRT) / FLOAT(VASP(JTH))**

**1500 GAP = MIN(GAP1,GAP2)**

**END**
*THIS SUBROUTINE CALCULATES THE STATISTICS FOR THE
LINK THAT THE VEHICLE HAS JUST EXITED. THE MIN TIME
DEPENDS ON THE TURNING SPEED RESTRICTION IN THE DISTCH
SUBROUTINE.*

**INCLUDE DEFINES**

**DIMENSION ICON(4)**

**DATA ICON / 0, 2, 6, 9 /**

**DEFINE ICONP(I) = ICON(I)**

**IF(KTHOLD .EQ. 0) GO TO 5**

**C-calculates time expected to travel link and
time vehicle should reach end of link.**

**LANE = VLAN(JTH)**

**IDELAY = CLOCK - VELKTM(JTH)**

**WRITE(21) CLOCK, KTHOLD, JTH, IDELAY**

**WRITE(6, 2) JTH, VELKTM(JTH), CLOCK, IDELAY**

**2 FORMAT(•###• IDELAY OUTPUT, 6I6)**

**LKVL(KTHOLD ,LANE) = LKVL(KTHOLD ,LANE) + 1**

**LKVD(KTHOLD ,TURN) = LKVD(KTHOLD ,TURN) + 1**

**LKDEL(KTHOLD ,LANE) = LKDEL(KTHOLD ,LANE) + IDELAY**

**LKDELD(KTHOLD ,TURN) = LKDELD(KTHOLD ,TURN) + IDELAY**

**C-calculates time expected to travel link and
time vehicle should reach end of link.**

**IF(IDSP .LT. VDSP(JTH)) IDSP = IDSP + 1**

**NUM = IDSP - VHOLD(JTH) - 1**

**ITURN = VTURN(JTH)**

**GO TO (10, 20, 30), TURN**

**C-left turn calculations depend on number of lanes.**

**MINTIM = (LKLNG(KTH) - NUM + ICONP(VDSP(JTH))) / VDSP(JTH)**

**GO TO 40**

**C-straight movements.**
20 MINTIM = (LKLNG(KTH) - NUM + LKLANS(KTH) * 2 + VDSP(JTH) - 1) / VDSP(JTH)  
   GO TO 40  
C  
C       RIGHT TURNS ARE NOT EFFECTED BY NUMBER OF LANES.  
C  
30 MINTIM = (LKLNG(KTH) - NUM - 1) / VDSP(JTH)  
40 VLKTIM(JTH) = CLOCK + MINTIM  
1 FORMAT('** WRITE(6,1) JTH,KTHOLD,KTH,TURN,LANE,DELAY,VLKTIM(JTH)  
C**** WRITE(6,1) JTH,KTHOLD,KTH,TURN,LANE,DELAY,VLKTIM(JTH)  
RETURN  
   END  
SUBROUTINE HOLDIN(JTH,IBTO,IBFROM,I)  
C  
C       THIS SUBROUTINE MOVES A VEHICLE INTO A HOLDING AREA.  
C       IT WILL BE REMOVED AND THE STATISTICS UPDATED BY  
C       THE HOLDOUT SUBROUTINE AFTER ALL LINK MOVEMENT  
C       HAS TAKEN PLACE.  
C  
INCLUDE DEFINS  
C  
BLKARY(IBTO) = JTH  
IF(IBFROM .NE. 0) BLKARY(IBFROM) = 0  
C  
C       VHOLD IS USED TO STORE THE NUMBER OF BLOCKS ALREADY MOVED.  
VHOLD(JTH) = 1  
C**** WRITE(6,1) JTH,IBTO,IBFROM,I  
1 FORMAT(' HOLDIN VEH **14** TO BLK **I4**, FROM BLK **I5**,  
1 " VHOLD **12")  
RETURN  
   END  
SUBROUTINE HOLDOUT  
C  
C       THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING  
C       AREA AND MOVES THEM ONTO THE LINK.  
C  
INCLUDE DEFINS  
DO 3000 KTH=1,NUMLKS  
LANE = 1  
C  
C       BRANCH ON NUMBER OF LANES  
IF(LKLANS(KTH) .EQ. 2) GO TO 2000  
C  
C       SINGLE LANE. LOOK AT HOLDING AREA. IF EMPTY SKIP.  
IB = LKFSTB(KTH) + LKLNG(KTH)  
IF(BLKARY(IB) .EQ. 0) GO TO 3000  
C  
C       IF NEW VEHICLE TO THIS LINK, ASSIGN PARAMETERS.  
JTH = BLKARY(IB)  
IF(VLK(JTH) .EQ. KTH) GO TO 1100  
KTHOLD = VLK(JTH)  
ITURN = VTURN(JTH)  
VLK(JTH) = KTH  
VLAN(JTH) = 1  
X = RN1(ISEED)  
DO 1050 I=1,2  
   IF(X .LE. LKPROB(KTH,I)) GO TO 1055
1050 CONTINUE
   VTUM(JTH) = 3
   GO TO 1075
1055 VTUM(JTH) = 1
1075 CALL GETSTA(JTH,KTHOLD,KTH,ITURN)
C
C       FIND LEADING VEHICLE.
1100 IBSTOP = LKFSTb(KTH)
   IBSTRT = IB - 1
   DO 1110 IBCHK = IBSTRT,IBSTOP,-1
   IF(BLKARY(IBCHK) NE. 0) GO TO 1200
1110 CONTINUE
C
C       NO LEADER ADVANCE DESIRFD NUMBER OF BLOCKS
   NUM = VASP(JTH)
   IF(NUM LT VDSP(JTH) .AND. VSTATE(JTH) NE. 1) NUM = NUM + 1
   IBNEW = IR - NUM + VHOLD(JTH)
   IF(IBNEW LT LKFSTb(KTH)) CALL DUMP(JTH,HOLDOU,*NoLEAd*,40)
   IF(IBNEW EQ. IB) GO TO 2975
   IF(BLKARY(IBNEW) NE. 0) CALL DUMP(JTH,HOLDOU,*FULLK*,42)
   GO TO 2950
C
C       LEADER FOUND.
1200 NUMPOS = IB - IBCHK - 1 + VHOLD(JTH)
   IF(NUMPOS LE. 0) GO TO 2975
   JTHLDR = BLKARY(IBCHK)
   CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
   NUMPOS = NUM - VHOLD(JTH)
   IF(NUMPOS LE. 0) GO TO 2975
   IBNEW = IR - NUMPOS
   GO TO 2950
C
C       TWO LANE LINK PROCESSING. LOOK AT HOLDING AREAS.
C
2000 IB = LKFSTb(KTH) + LKLNG(KTH) * 2
   IF(BLKARY(IB) NE. 0) GO TO 2100
2050 IB = IB + 1
   LANE = LANE + 1
   IF(BLKARY(IB) NE. 0) GO TO 2100
   IF(LANE GE. LKLANS(KTH)) GO TO 3000
   GO TO 2050
C
C       FOUND A VEHICLE IN THE HOLDING AREA. IF -VEHICLE
C       NEW TO THIS LINK, ASSIGN PARAMETERS.
C
2100 JTH = BLKARY(IB)
   IF(VLK(JTH) EQ. KTH) GO TO 2150
   KTHOLD = VLK(JTH)
   ITURN = VTUM(JTH)
   VLK(JTH) = KTH
   X = RN(ISEED)
   IF(X GT. LKPROP(KTH+1)) GO TO 2140
   VTUM(JTH) = 1
   VLAN(JTH) = 1
   GO TO 2149
2140 IF(X .GT. LKPROB(KTH,2)) GO TO 2145
    IF(VTURN(JTH),EQ.2) GO TO 2149
    VTURN(JTH) = 2
    VLAN(JTH) = RNK(ISEED) * LKLANS(KTH) + 1
    GO TO 2149
2145 VTURN(JTH) = 3
    VLAN(JTH) = LKLANS(KTH)
2149 CALL GETSTA(JTH,KTH,ITURN)
C
C ***** FIND LEADER IN DESIRED LANE.
C
2150 IBSTOP = LKFSTB(KTH) + LKLNG(KTH) * (VLAN(JTH) - 1)
    IBSTRT = IBSTOP + LKLNG(KTH) - 1
    DO 2160 IBCHK=IBSTRT,IBSTOP,-1
        IF(BLKARY(IBCHK) .NE. 0) GO TO 2175
    2160 CONTINUE
C
C NO LEADER FOUND ADVANCE
NUM = VASP(JTH)
    IF(NUM .LT. VASP(JTH) .AND. VSTATE(JTH) .NE. 1) NUM = NUM + 1
    IBNEW = IBSTRT + 1 - NUM + VHOLD(JTH)
    IF(IBNEW .LT. IBSTOP) IBNEW = IBSTOP
    IF(IBNEW .GT. IBSTRT) GO TO 2975
    IF(BLKARY(IBNEW) .NE. 0) CALL NUMP(JTH,'HOLDOU','FULRLK',107)
    GO TO 2950
C
C LEADER FOUND
2175 NUMPOS = IBSTRT - IBCHK + VHOLD(JTH)
    IF(NUMPOS .LE. 0) GO TO 2975
    JTHLDR = RLKARY(IBCHK)
    CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
    NUMPOS = NUM - VHOLD(JTH)
    IF(NUMPOS .LE. 0) GO TO 2975
    IBNEW = IBSTRT + 1 - NUMPOS
    GO TO 2950
C
C ***** MOVE VEHICLE FORWARD NUM BLOCKS
C
2950 CONTINUE
    VHOLD(JTH) = 0
    BLKARY(IBNEW) = JTH
    BLKARY(IB) = 0
    CALL UPDATV(JTH,NUM,1)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
    IF(LANE .LT. LKLANS(KTH)) GO TO 2050
    GO TO 3000
C
C ***** VEHICLE CAN NOT MOVE OUT OF HOLDING AREA.
C
2975 CONTINUE
    VHOLD(JTH) = 0
    CALL UPDATV(JTH,0,1)
    NZERO = 0
C**** WRITE(6,1) JTH,IB,IB,NZERO
    1 FORMAT(' HOLDOU','416)
    IF(LANE .LT. LKLANS(KTH)) GO TO 2050
3000 CONTINUE
FUNCTION IBHOLD(KTH)

   RETURN
END

FUNCTION IBHOLD(KTH)

   . . . . THIS FUNCTION SEARCHES THE HOLDING AREA OF THE
   KTH LINK. IF THERE IS AN EMPTY SPACE IN THE
   HOLDING AREA, THE BLOCK ADDRESS IS RETURNED AS
   THE VALUE OF THE FUNCTION. IF THE AREA IS FULL,
   ZERO IS THE RETURNED VALUE OF THE FUNCTION.

INCLUDE DEFINS

   . . . . BRANCH BASED ON THE NUMBER OF LANES.
   IBHOLD = 0
   IF(LKLANS(KTH) .EQ. 2) GO TO 1000

   . . . . SINGLE LANE; IF HOLDING AREA FULL THEN RETURN.
   IB = LKFSTB(KTH) + LKLNG(KTH)
   IF(BLKARY(IB) .NE. 0) RETURN
   IBHOLD = IB
   RETURN

   . . . . TWO LANES - MUST CHECK BOTH HOLDING BLOCKS.
   1000 IB = LKFSTB(KTH) + LKLNG(KTH) * 2
   IF(BLKARY(IB) .NE. 0) GO TO 1010
   IBHOLD = IB
   RETURN

   1010 IB = IB + 1
   IF(BLKARY(IB) .NE. 0) RETURN
   IBHOLD = IB
   RETURN
END

SUBROUTINE INPLKS

   . . . . THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINES.
   AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
   CALCULATES SOME CROSS REFERENCE INDICES.

INCLUDE DEFINS

   . . . . THIS IS THE LOCAL DECLARATION.

INTEGER ALPHA
KTH=0
1000 KTH=KTH+1
   READ(5,101) ALPHA,LKIN(KTH+1),LKID(KTH+2),LKLNG(KTH),LKLANS(KTH),
   LKOPOS(KTH),LKPPOS(KTH+1),LKPPOS(KTH+2),(LKDEST(KTH+K),K=1,3)

   . . . . IF IT IS A BLANK CARD, GO TO 2000.
   IF(ALPHA .EQ. 1H ) GO TO 2000

   . . . . CALCULATE THE BLOCKS PER LANE AND THE NUMBER OF
   BLOCKS USED THUS FAR.
   LKLNG(KTH)=LKLNG(KTH)/LENGTH
   LKFSTB(KTH)=NRLKS+1
   NBLKS=NBLKS+LKLNG(KTH)*LKLANS(KTH)+LKLANS(KTH)
C ... FIND NODE NUMBER OF THE HEAD NODE.

DO 1010 NTH=1,NUMNOS
   IF(NOID(NTH) .EQ. LKID(KTH)) GO TO 1015
1010 CONTINUE
   CALL DUMP(KTH,'INPLKS','D01010',1)
1015 LKNOD(KTH)=NTH
C ...
C SET UP THE TURNING PROBABILITIES.

1060 LKPROB(KTH,2)=LKPROB(KTH,1)+LKPROB(KTH,2)
   IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH,2) = 1.0
   GO TO 1000
C ...
C FIX THE NUMBER OF LINKS IN NETWORK.

2000 NUMLKS=KTH-1
   IF(NUMLKS .GE. MAXLKS) CALL NIMP(NUMLKS,'INPLKS','MAXLKS',1)
   IF(NBLKS .GT. MAXRLK) CALL DIMP(NBLKS,'INPLKS','MAXRLK',1)
C ...
C FIND THE LINK OPPOSING A LEFT TURN.

DE 2500 KTH=1,NUMLKS
   IF(LKPOS(KTH) .EQ. 0) GO TO 2500
   DO 2400 KTH2=1,NUMLKS
      IF(LKID(KTH2) .NE. LKID(KTH)) GO TO 2400
      IF(LKPOS(KTH) .EQ. LKID(KTH2)) GO TO 2450
2400 CONTINUE
   CALL DUMP(KTH,'INPLKS','LKPOS',1)
2450 LKPOS(KTH)=KTH2
2500 CONTINUE
RETURN
101 FORMAT(A1,2I3,I4,H,I3,2F3.3,I3)
END

SUBROUTINE INTERL
C ...
C THIS SUBROUTINE CALLS THE HANDLER APPROPRIATE TO
C THE NODE TYPE.

INCLUDE DEFS
DO 2000 NTH=1,NUMNOS
C ...
C IF EXTERNAL NODE FORGET IT AND GO TO THE NEXT ONE.

IF(NOTYPE(NTH) .EQ. 2) GO TO 2000
C ...
C SET UP A COMPUTED GO TO BASED ON NODE GEOMETRY.

K=NOGEOM(NTH)
   GO TO (1100,1200,1300)*K
C ...
C TWO-BY-TWO.
1100 CALL MOVETT(NTH)
   GO TO 2000
C ...
C FOUR-BY-TWO.
1200 CALL DUMP(NTH,'INTERL','4X2','23)
GO TO 2000

C C ..... FOUR-BY-FOUR.
C 1300 CALL MOVEFF(NTH)
C 2000 CONTINUE
C RETURN
C END
C SUBROUTINE INTERN(NTH)
C C ..... THIS SUBROUTINE READS THE DATA CARDS ON INTERIOR
C NODES ONLY. CONTROL IS RETURNED TO THE CALLING
C PROGRAM AFTER A BLANK CARD IS FOUND.
C
INTEGER ALPHA
INCLUDE DEFIN

ITH=0
1000 NTH=NTH+1
READ(5,101) ALPHA,NODNO(NTH),NOTYPE(NTH),(NONXT(NTH,K),K=1,4),
1 NOGEOM(NTH)

C IF(ALPHA .EQ. 1H ) GO TO 3000
C
C ..... PORTION OFF A SECTION OF BLOCKS FOR THE INTERSECTION.
C NOFSTB(NTH)=NRLKS+1
C K=NOGEOM(NTH)
C GO TO (1100,1200,1300)*K
1100 NBLKS=NBLKS+4
GO TO 1900
1200 NBLKS=NBLKS+8
GO TO 1900
1300 NBLKS=NBLKS+16
1900 CONTINUE
C IF(NOTYPE(NTH) .LT. U) GO TO 1000
C
C ..... SIGNALIZED INTERSECTIONS NEED TO READ ANOTHER DATA
C CARD TO GET CONTROL INFORMATION.
C ITH=ITH+1
C NOSIG(NTH)=ITH
READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
1 SIGOFF(ITH,2),SIGGRN(ITH,2)

C C ..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C DO 2100 I=1,2
C SIGRED(IPTH,1) = SIGCYC(ITH) - SIGGRN(IPTH,1) - AMBERT
C TEMP = SIGOFF(IPTH,1) + SIGGRN(IPTH,1)
C IF(TEMP .GT. SIGCYC(ITH)) GO TO 2010
C TEMP = TEMP + AMBERT
C IF(TEMP .GT. SIGCYC(ITH)) GO TO 2005
C SIGSTA(IPTH,1) = 2
C SIGCLK(IPTH,1) = SIGOFF(IPTH,1)
C GO TO 2100
2005 SIGSTA(IPTH,1) = 1
C SIGCLK(IPTH,1) = TFMP - SIGCYC(ITH)
GO TO 2100
2010 SIGSTA(IPTH,1) = 0
C SIGCLK(IPTH,1) = TFMP - SIGCYC(ITH)
GO TO 2100
2100 CONTINUE
GO TO 1000
3000 NUMNOS = NTH - 1
   IF(NUMNOS .GE. MAxNO) CALL DUMP(NUMNOS,'INTERN','MAXNO',1)
   NUMSIG = ITH
   IF(NUMSIG .GE. MAXSIG) CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
RETURN
C
C  ..... FORMAT STATEMENTS
C
101 FORMAT(A1,I3,I2,u13,I3,I1)
102 FORMAT(A1,I3,I5,I3)
END
SUBROUTINE INTRAL
C
C  THIS SUBROUTINE CHECKS EACH LINK FOR MOVEMENT
C  WITHIN THE LINK.
C
INCLUDE DEFINS
DIMENSION NUADD(4,2),ISQ2(4,2,4)
DATA ISQ2 /9*10,11,12,13,14,15,16, 15,11,7,3,16,12,8,4, 1
8,7,6,5,4,3,2,1, 2,6,10,14,1,5,9,13/
DATA NUADD /3,4,2,1,4,2,1,3/
INTEGER POS
C
C  ..... SEARCH EACH LINK IN NETWORK.
C
DO 6000 KTH=1,NUMLKS
   LANE = 0
   NTH = LKNOD(KTH)
   ITH = NOSIG(NTH)
C
C  ..... FIND THE ARM THE VEHICLE IS ON.
C
   DO 560 IARM=1,4
      IF(NOINP(NTH,IARM) .EQ. KTH) GO TO 561
   560 CONTINUE
      CALL DUMP(JTH,'INTRAL','ARM'+27)
   561 CONTINUE
   1000 LANE = LANE + 1
C
C  ..... SET UP THE BLOCK LIMITS FOR THIS LANE.
C
   IBSTRT = LKFSTB(KTH) + (LANE - 1) * LKLNG(KTH)
   IBSTOP = IBSTRT + LKLNG(KTH) - 1
   POS = 0
   JTHLDR = 0
C
C  ..... STEP THROUGH EACH BLOCK ON THE LINK.
C
   DO 5000 IB=IBSTRT,IBSTOP
      POS = POS + 1
      IF(IBKARY(IB) .EQ. 0) GO TO 5000
   5000 CONTINUE
JTH = BLKARY(IB)
NUM1 = VASP(JTH)
IF(NUM1 .LT. VUSP(JTH) .AND. VSTATE(JTH) .NE. 1) NUM1 = NUM1 + 1
CALL DISTCH(JTH,KTH,POS+NTH+NUM2,I01,1,0,TA1)
NUM = MIN(NUM1,NUM2)
IF(JTHLDR .EQ. 0) GO TO 1525

C FOR FOLLOWERS ONLY.
C
1500 NUMPOS = IB - IBLDR - 1
CALL ADVCHK(JTH,JTHLDR,NUMPOS,NUM)
C
BOTH LEADERS AND FOLLOWERS.
C
1525 IF(NUM .EQ. 0) GO TO 4990
IF(VSTATE(JTH) .EQ. 0) GO TO 4980
IF(POS = NUM .GE. 1) GO TO 4975
C
VEHICLE CAN MOVE, BUT THE INTERSECTION IS INVOLVED. BRANCH ON INTERSECTION GEOMETRY.
C
IF(NOTYPE(NTH) .LT. 2) GO TO 1800
IF(NOGEOM(NTH) .EQ. 3) GO TO 2000
C
A TWO-BY-TWO INTERSECTION IS INVOLVED.
C
NUMDS = NUM - POS + 1
C
BRANCH TO SECTION WITH PROPER DESTINATION.
C
1561 IF(NUMDS .EQ. 1) GO TO 1800
IF(VTURN(JTH) .NE. 3) GO TO 1580
I = 3
NUM = POS
GO TO 1850
1580 IF(NUMDS .EQ. 2) GO TO 1825
I = VTURN(JTH)
NUM = POS + 1
GO TO 1850
C
MOVE FORWARD ONE SQUARE INTO THE INTERSECTION.
C
1800 IBNEW = NOFSTD(NTH) + NUADD(IARM,1) - 1
GO TO 4971
C
MOVE FORWARD TWO SQUARES INTO THE INTERSECTION.
C
1825 IBNEW = NOFSTD(NTH) + NUADD(IARM,2) - 1
GO TO 4971
C
ADVANCE INTO THE HOLDING AREA.
C
1850 KTHNEW = LKDEST(KTH,I)
IBNEW = IRIHOLD(KTHNEW)
IF(IBNEW .EQ. 0) CALL DUMP(JTH,'INTERL','IBHOLD',105)
CALL HOLDIN(JTH,IRNEW,IB,NUM)
GO TO 5000
***** A FOURS BY FOUR INTERSECTION IS INVOLVED.

2000 NUMDS = NUM - POS + 1
IF(NUMDS .NE. 1) GO TO 2100

***** MOVE FORWARD ONE SQUARE INTO INTERSECTION.

IBNEW = NOFSTR(NTH) - 1 + ISQ?(LANE, IARM)
GO TO 4971

***** MORE THAN ONE SQUARE.

2100 IF(VTURNC(JTH) .NE. 2) CALL D1MP(JTH, INTRAL, TURNSP, 126)
IF(NUMDS .GT. 4) GO TO 2200
2150 IBNEW = NOFSTR(NTH) - 1 + ISQ?(NUMDS, LANE, IARM)
GO TO 4971

***** TRY TO CROSS INTERSECTION INTO HOLDING AREA.

2200 IBNEW = INHOLD(LKNEST(KTH*2))
IF(IBNEW .NE. 0) GO TO 2250
I = NUMDS - 4
NUMDS = 4
NUM = NUM - I
GO TO 2150
2250 CALL HOLDIN(JTH, IBNEW, IB, NUM)
GO TO 5000

***** VEHICLE IS GOING TO A TERMINATE NODE.

4970 CALL TERMIN(JTH, IR)
GO TO 5000

***** VEHICLE IS GOING INTO THE INTERSECTION.

4971 BLKARY(IB) = 0
BLKARY(IBNEW) = JTH
CALL UPDATV(JTH, NUM, 1)

C*** WRITE(6, 1) JTH, IR, IBNEW, NUM
GO TO 5000

***** SIMPLE MOVE OF *NUM* BLOCKS ON SAME LINK.

4975 IBNEW = IB - NUM
BLKARY(IB) = 0
BLKARY(IBNEW) = JTH
CALL UPDATV(JTH, NUM, 1)

C*** WRITE(6, 1) JTH, IR, IBNEW, NUM
GO TO 4995

***** HESITATE THIS TIME. CAN MOVE NEXT TIME.

4980 VSTATE(JTH) = 3
IBNEW = IB
CALL UPDATV(JTH, 0, 1)
153

NZERO = 0
C**** WRITE(6,1) JTH,IB,IB,NZERO
GO TO 4995
C
C
(...) SIT STILL AND WAIT.
C
4990 IBNEW = IB
CALL UPDATV(JTH,0,0)
C**** WRITE(6,1) JTH,IB,IB,NZERO
1 FORMAT(5,INTERL,4I6)
C
C
(...) MAKE THIS VEHICLE THE CURRENT LEADER.
C
4995 JTHLDR = JTH
IBLDR = IBNEW
C
C
(...) RETURN TO TOP AND LOOK AT NEXT BLOCK.
C
5000 CONTINUE
C
C
(...) ANOTHER LANE*
C
IF(LANE .LT, LKLANS(KTH)) GO TO 1000
6000 CONTINUE
RETURN
END
SUBROUTINE MOVEFF(NTH)
C
C THIS SUBROUTINE MOVES VEHICLES IN A 4X4 INTERSECTION.
C THE RED OR AMBER VEHICLES ARE CHECKED FIRST, THEN
C THOSE FACING A GREEN LIGHT.
C
INCLUDE DEFINES
INTEGER ORDER,LEFTSQ,POS,STRTSQ
DIMENSION ORDER(4,2),LEFTSQ(5,4),STRTSQ(4,2,4)
DATA ORDER /1,3,2,4,2,4,1,3 /
DATA LEFTSQ /1,7,11,4,8, 5,6,7,1,2, 14,10,6,13,9, 12,11,10,16,15 /
DATA STRTSQ /9,19,11,12,13,14,15,16, 15,11,7,13,16,12,8,4
1 8,7,6,5,4,3,2,1, 2,6,10,14,15,9,13 /
C
C
(...) SETUP TO GET RED AND AMBER FIRST.
C
ITH = NOSIG(NTH)
10 IF(SIGSTA(ITH,1)-1) 10,20,30
J = 2
60 TO 50
20 J = 1
60 TO 50
30 IF(SIGSTA(ITH,2)-2) 20,10,20
50 IBASE = NOFSTA(NTH) - 1
C
C
(...) TAKE EACH APPROACH LINK AND LOOK FOR CARS FROM
C
THAT LINK IN THE INTERSECTION.
C
DO 1000 I=1,4
IARM = ORDER(I,J)
KTH = NOINP(NTH,IARM)
***** FIRST GET RIGHT TURNS

IB = IBASE + STRT5Q(1,2,1ARM)
IF(BLKARY(IB) .EQ. 0) GO TO 110
JTH = BLKARY(IB)
IF(VLK(JTH) .NE. KTH OR VTURN(JTH) .NE. 3) GO TO 110
IBNEW = IBHOLnEST(KTH,1))
IF(IBNEW .EQ. 0) GO TO 75
CALL HOLDIN(JTH,IBNEW,IB,0)
GO TO 100
75 CALL UPDATV(JTH,0,0)
NO = 0
C**** WRITE(6,1) JTH,IB,IB,NO
1 FORMAT(• MOVEFF•4IS)
C
***** NEXT LOOK AT LEFT TURNS WHICH HAVE BEGUN TO MOVE
C OR ARE READY TO START.
C
100 LPOS = 0
C
***** FIRST SQUARE.
C
IB = IBASE + LEFT5Q(1,1ARM)
IF(BLKARY(IB) .EQ. 0) GO TO 110
JTH = BLKARY(IB)
IF(VLK(JTH) .NE. KTH) GO TO 107
IBNEW = IBHOLnEST(KTH,1))
IF(IBNEW .EQ. 0) GO TO 105
CALL HOLDIN(JTH,IBNEW,IB,0)
GO TO 130
105 CALL UPDATV(JTH,0,0)
NO = 0
C**** WRITE(6,1) JTH,IB,IB,NO
LPOS = 1
GO TO 130
107 LPOS = 1
C
***** CHECK SECOND SQUARE.
C
110 IB = IBASE + LEFT5Q(2,1ARM)
IF(BLKARY(IB) .EQ. 0) GO TO 130
JTH = BLKARY(IB)
IF(VLK(JTH) .NE. KTH) GO TO 125
IF(LPOS .EQ. 1) GO TO 105
IF(VASP(JTH) .GT. 0) GO TO 120
115 IBNEW = IBASE + LFFTSQ(1,1ARM)
BLKARY(IB) = 0
BLKARY(IBNEW) = JTH
CALL UPDATV(JTH,1,0)
N1 = 1
C**** WRITE(6,1) JTH,IB,IBNEW,N1
LPOS = 1
GO TO 130
120 IBNEW = IBHOLnEST(KTH,1))
IF(IBNEW .EQ. 0) GO TO 115
CALL HOLDIN(JTH,IBNEW,IB,1)
GO TO 130

125 LPOS = 1

C

***** NOW CHECK TURNING POINT.

130 IB = IBASE + LEFTSQ(3, IARM)
IF(BLKARY(IB) .EQ. 0) GO TO 150
JTH = BLKARY(IB)
IF(VLK(JTH) .NE. KTH .OR. VTUM(JTH) .NE. 1) GO TO 140
IF(LPOS .EQ. 1) GO TO 145
CALL GAPCHK(LKOPOS(KTH), KOUT, NTH, ITH, 2-MOD(IARM, 2))
IF(KOUT .EQ. 1) GO TO 145

C

***** CHECK FIRST TWO SQUARES OF INTERSECTION FOR TRAFFIC FROM OPPOSING LINK.

IBCHK = IBASE + LEFTSQ(4, IARM)
JTHCHK = BLKARY(IBCHK)
IF(JTHCHK .EQ. 0) GO TO 140
IF(VLK(JTHCHK) .EQ. LKOPOS(KTH)) GO TO 145

140 IBCHK = IBASE + LEFTSQ(5, IARM)
JTHCHK = BLKARY(IBCHK)
IF(JTHCHK .EQ. 0) GO TO 142
IF(VLK(JTHCHK) .EQ. LKOPOS(KTH)) GO TO 145

C

***** OK TO MOVE VEHICLE INTO TURNING AREA.

142 NUM = VASP(JTH) + 1
IBNEW = IBASE + LEFTSQ(3-NUM, IARM)
BLKARY(IB) = 0
BLKARY(IBNEW) = JTH
C**** WRITE(6,1) JTH, IB, IBNEW, NUM
CALL UPDATV(JTH, NUM, 0)
GO TO 150

145 CALL UPDATV(JTH, 0, 0)
NO = 0
C**** WRITE(6,1) JTH, IB, IB, NO
150 CONTINUE

C

***** LEFT IN MAIN FLOW TREATED WITH STRAIGHT TRAFFIC.

DO 400 LANE = 1, 2
JTHL = 0
DO 400 POS = 4, 1, -1
IB = IBASE + STRT5W(POS, LANE, IARM)
IF(BLKARY(IB) .EQ. 0) GO TO 400
JTH = BLKARY(IB)
IF(VLK(JTH) .NE. KTH) GO TO 390
IF(JTHL) 380, 210, 250

C

***** NO LEADER FOUND.

210 IF(VTUM(JTH) .EQ. 2) 211, 215, 400

C

***** LEFT TURN VEHICLES.

211 IF(POS .EQ. 3) GO TO 212
NUM = 1
GO TO 220

212 JTHL = JTH
LPOS = POS
GO TO 400

C

***** STRAIGHT VEHICLES
C

215 NUM = VASP(JTH)
IF(NUM .LT. VDSP(JTH) .AND. VSTATE(JTH) .NE. 1) NUM = NUM + 1
IF(NUM+POS .LT. 4) GO TO 225

220 IBNEW = IBASE + STRTSQ(POS+NUM,LANE,IARM)
BLKARY(IB) = 0
BLKARY(IBNEW) = JTH
CALL UPDATV(JTH,NUM,0)
C**** WRITE(6,1) JTH,IB,IBNEW,NUM
JTHL = JTH
LPOS = POS
GO TO 400

225 IBNEW = IBHOLD(LKNEST(KTH-2))
IF(IBNEW .NE. 0) GO TO 230
NUM = 4 - POS
GO TO 220

230 CALL HOLDIN(JTH,IBNEW,IB,4-POS)
GO TO 400

C

***** LEADER IS PRESENT
C

250 NGAP = LPOS - POS - 1
IF(NGAP .EQ. 0) GO TO 380
IF(VTURN(JTH)-2) 211,260*400

260 CALL ADVCHK(JTH,JTHL,NGAP,NUM)
IF(NUM .EQ. 0) GO TO 380
GO TO 220

380 JTHL = JTH
LPOS = POS
NO = 0
C**** WRITE(6,1) JTH,IB,IB,NO
CALL UPDATV(JTH,0,0)
GO TO 400

C

***** LEADER IS PRESENT FROM ANOTHER LINK
C

390 LPOS = POS
JTHL = -1
400 CONTINUE
1000 CONTINUE
RETURN
END

SUBROUTINE MOVETT(NTH)

C

C THIS SUBROUTINE MOVES VEHICLES WHICH ARE IN A
C TWO BY TWO INTERSECTION. YOU ARE LOOKING AT
C THE NORTH WST SQUARE OF THE INTERSECTION AND
C THE ARMS ARE LABELED AS SHOWN BELOW:
C


INCLUDE DEFINS

***** LOCAL ARRAYS

DIMENSION IARM(4,4),IR13(4)
DATA IB13/ 2,1,1,-2 /
DATA IARM/1,4,2,3,2,1,3,4,3,2,4,1,4,3,1,2 /

***** SET UP A BASE FOR REFERENCING THE BLOCK ARRAY.
IBASE = NOFSTR(NTH) - 1

***** CONSIDER EACH SQUARE OF INTERSECTION
DO 2000 ISQUA = 1,4

***** FIND OUT IF THERE IS A VEHICLE IN THAT SQUARE, AND IF NOT GO TO NEXT SQUARE.
IB = IBASE + ISQUA
JTH = BLKARY(IB)
IF (JTH .EQ. N) GO TO 2000
IF (UMOVE(JTH) .EQ. 1) GO TO 2000

***** FIND OUT WHICH ARM VEHICLE IS FROM AND BRANCH.
KTHOLD = VLK(JTH)
I3ARM = IARM(ISQUA*3)
IF (NOINP(NTH,I3ARM) .EQ. KTHOLD) GO TO 1001
I4ARM = IARM(ISQUA*4)
IF (NOINP(NTH,I4ARM) .EQ. KTHOLD) GO TO 1500
I2ARM = IARM(ISQUA*2)
IF (NOINP(NTH,I2ARM) .EQ. KTHOLD) GO TO 1300

***** IF THESE TESTS FAIL AN ERROR HAS OCCURRED
CALL DUMP(JTH,'MOVE','NOINP','57)

***** VEHICLE HAS COME FROM INPUT 3
1001 IF (VTURN(JTH) .EQ. 1) GO TO 1200

***** INPUT 3 STRAIGHT TRAFFIC
KTHNEW = LKDEST(KTHOLD*2)
***** CHECK HOLDING SPACE
IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)

***** IF HOLDING AREA NOT AVAILABLE, GO TO 1975.
IF(BLKARY(IBNEW) .NE. 0) GO TO 1975

***** OTHERWISE MOVE TO HOLDING AREA.
NOMOVE = 0
GO TO 1960

INPUT 3 LEFT TURN. IF SQUARE 3 NON EMPTY, NO MOVE POSSIBLE.
1200 IBCHK = IR13(ISQUAR) + IB
IF(BLKARY(IBCHK) .NE. 0) GO TO 1975

***** NEXT CHECK APPOSING TRAFFIC. SKIP CHECK IF NO LINK.
KTHPO = LKOPOS(KTHOLD)
IF(KTHPO .EQ. 0) GO TO 1250
CALL GAPCHK(KTHPO,KOUT,NTH,NTHSIG(NTH),2-MOD(I3ARM,2))

***** IF GAP NOT RIG ENOUGH, WAIT
IF(KOUT .NE. 0) GO TO 1975

***** NEXT CHECK HOLDING AREA.
1250 KTHNEW = LKDEST(KTHOLD+1)
IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)

***** IF HOLDING AREA NOT AVAILABLE, WAIT
IF(BLKARY(IBNEW) .NE. 0) GO TO 1975

***** OK TO TURN. ADVANCE TO SQUARE 3.
GO TO 1965

****** VEHICLE HAS COME FROM INPUT 2 AND IS TURNING LEFT.
1300 KTHNEW = LKDEST(KTHOLD+1)

****** CHECK HOLDING AREA.
IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)

****** IF HOLDING AREA BLOCKED, WAIT.
IF(BLKARY(IBNEW) .NE. 0) GO TO 1975

****** OTHERWISE MOVE TO HOLDING AREA.
NOMOVE = 0
GO TO 1960

****** VEHICLE HAS COME FROM INPUT 4. RANCH ON TURNING MOVEMENT.
1500 K = VTURN(JTH)
GO TO (1700,1600,1550) + K

****** INPUT 4 RIGHT TURN. CHECK HOLDING AREA
1550 KTHNEW = LKDEST(KTHOLD+3)
IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)
C ..... IF AREA BLOCKED, WAIT.*
   IF(BLKARY(IBNEW).NE.0) GO TO 1975
C
C ..... MOVE TO HOLDING AREA.*
   NOMOVE = 0
   GO TO 1960
C
C ..... INPUT 4 STRAIGHT MOVEMENT. CHECK SQUARE 3 AND SPEED.*
1600 IBCHK = IB13(ISQUAR) + IB
   IF(BLKARY(IBCHK).NE.0) GO TO 1975
   IF(VASP(JTH).LT.1) GO TO 1965
C
C ..... NEXT CHECK HOLDING AREA.*
   KTHNEW = LKDEST(KTOTH*2)
   IBNEW = LKFSTR(KTHNEW) + LKLNG(KTHNEW)
C
C ..... IF AREA BLOCKED, ADVANCE TO SQUARE 3
   IF(BLKARY(IBNEW).NE.0) GO TO 1965
C
C ..... MOVE VEHICLE THROUGH ONE BLOCK TO HOLDING AREA.*
   NOMOVE = 1
   GO TO 1960
C
C ..... INPUT 4 LEFT TURN. CHECK SQUARE 3.*
1700 IBCHK = IR13(TSQUAR) + IB
   IF(BLKARY(IBCHK).NE.0) GO TO 1975
   GO TO 1965
C
C ..... OK TO MOVE VEHICLE TO HOLDING AREA OF DESTINATION.*
C
1960 CALL HOLDIN(JTH, IBNEW, IB, NOMOVE)
   GO TO 2000
C
C ..... ADVANCE ONE SQUARE COUNTERCLOCKWISE.*
C
1965 IBNEW = IB + IB13(ISQUAR)
   BLKARY(IB) = 0
   BLKARY(IBNEW) = JTH
   CALL UPDATV(JTH, 1, 0)
   N1 = 1
C**** WRITE(6, 1) JTH, IB, IBNEW, N1
   GO TO 2000
C
C ..... NO MOVEMENT WAS POSSIBLE.*
C
1975 CALL UPDATV(JTH, 0, 0)
   NO = 0
C**** WRITE(6, 1) JTH, IB, IB, NO
   FORMAT(* MOVETT'*, 16)
2000 CONTINUE
   RETURN
END
FUNCTION RN1(ISEEN)
C
C ..... THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
C RANDOM NUMBERS BETWEEN 0 AND 1
ISEED=ISEED$18533$
IF(ISEED .LT. 0) ISEED=ISEED$34359738367+1$
RN1=ISEED$2.0**(-35)$
RETURN
END

SUBROUTINE SETUP

***** THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
FILES BEFORE THE SIMULATION BEGINS.

INCLUDE DEFINS

**** FOR EACH NODE
DO 2000 NTH=1,NUMNOS
IF(N0TYPE(NTH) .GT. 2) GO TO isOO
IF(N0TYPE(NTH) .EQ. 2) GO TO 1300

**** GENERATE NONES, SET TIME OF FIRST ARRIVAL.
NOPARS(NTH+1) = 3600. / NOPARS(NTH+1)
DO 1001 Irl,100
1001 A = RN1(ISEED)
NOSEED(NTH) = ISEED
NOCLK(NTH) = TBAGFN(NTH)

***** FIND LINK TO PUT VEHICLES ON.
DO 1100 KTH=1,NUMLKS
IF(NONXT(NTH+1) .EQ. LKID(KTH+2) .AND. NOIDNO(NTH) .EQ. LKID(KTH+1)) GO TO 1110
1100 CONTINUE
CALL DUMP(KTH,'SETUP '•'DO1000••1>
1110 NOOUTP(NTH) = KTH

***** IF GENERATE ONLY SKIP TO NEXT NODE.
IF(N0TYPE(NTH) .EQ. 1) GO TO 2000

***** FIND INDEX TO LINK POINTING AT THIS NODE.
1300 CONTINUE
DO 1350 KTH=1,NUMLKS
IF(NONXT(NTH+1) .EQ. LKID(KTH+1) .AND. NOIDNO(NTH) .EQ. LKID(KTH+2)) 11)
1350 CONTINUE
CALL DUMP(KTH,'SETUP '•'DO1350••1>
1360 NOINP(NTH+1) = KTH
GO TO 2000

***** INTERNAL NODES
1500 CONTINUE
IDNTH = NOIDNO(NTH)
ITH = NOSIG(NTH)

***** TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
NODE.
DO 1600 KTH=1,NUMLKS
C C CHECK FOR INPUT TO THIS NODE.
I F (L K I D ( K T H , 2 ) . N E . I N N T H ) G O T O 1 6 0 0
C C OK, NOW FIND OUT WHERE IT CAME FROM.
D O 1 5 2 0 I = 1 , 4
I F ( L K I D ( K T H , 1 ) . E Q . N O N X T ( N T H , I ) ) G O T O 1 5 2 5
1 5 2 0 C O N T I N U E
C A L L D U M P ( N T H , ’ S E T U P ’ ) ’ D 0 1 5 2 0 ’ I
1 5 2 5 N O I N P ( N T H , I ) = K T H
1 6 0 0 C O N T I N U E
2 0 0 0 C O N T I N U E
C C FOR EACH LINK FIND THE DESTINATIONS.
D O 3 0 0 0 K T H = 1 , N U M _ L K S
D O 2 5 0 0 I = 1 , 3
I F ( L K D E S T ( K T H , I ) . E Q . 0 ) G O T O 2 5 0 0
D O 2 4 0 0 K T H C H K = 1 + N U M _ L K S
I F ( L K I D ( K T H C H K , 2 ) . N E . L K D E S T ( K T H , I ) ) G O T O 2 4 0 0
I F ( L K I D ( K T H C H K , 1 ) . N E . L K I D ( K T H , 2 ) ) G O T O 2 4 0 0
L K D E S T ( K T H , I ) = K T H C H K
G O T O 2 5 0 0
2 4 0 0 C O N T I N U E
2 5 0 0 C O N T I N U E
3 0 0 0 C O N T I N U E
R E T U R N
E N D
S U B R O U T I N E S I G C H K
C C THIS SUBROUTINE CHECKS THE SIGNAL SETTINGS TO SEE IF A CHANGE IS NECESSARY.
C I N C L U D E D E F I N S
D O 2 0 0 0 I T H = 1 , N U M _ S I G
D O 1 9 9 9 I = 1 , 2
C C FIRST CHECK THE SIGNAL’S CLOCK.
C I F ( S I G C L K ( I T H , I ) . G T . C L O C K ) G O T O 1 9 9 9
C C BRANCH BASED ON OLD STATE.
C I F ( S I G S T A ( I T H , I ) = 1 ) 1 1 0 0 , 1 2 0 0 , 1 3 0 0
C C OLD STATE WAS GREEN, CHANGE TO AMBER.
1 1 0 0 S I G S T A ( I T H , I ) = 1
G O T O 1 9 9 9
C C OLD STATE WAS AMBER, CHANGE TO RED.
1 2 0 0 S I G S T A ( I T H , I ) = 2
G O T O 1 9 9 9
C C OLD STATE WAS RED, CHANGE TO GREEN.
1 3 0 0 S I G S T A ( I T H , I ) = 0
SIGCLK(I1*I2*I3*I4) = SIGCLK(I1*I2*I3) + SIGGRN(I1*I2*I3)

***** CHECK QUEUE LENGTHS ON INPUTS WHICH WERE RED.

DO 1500 Ip=1,2
II = I + 2 * (IP - 1)
KTH = SIGINP(I1*I2*I3)

***** CHECK FIRST LANE FOR QUEUE.

IB1 = LKFSTA(KTH)
IB2 = IB1 + LKLNG(KTH) -1
LQ = 0
DO 1420 IR=IB2,IB1,-1
JTH = BLKARY(IB)
IF(JTH .EQ. 0) GO TO 1420
IF(LQ .NE. 0) GO TO 1410
IF(UASP(JTH) .NE. 0) GO TO 1420
1410 LQ = LQ + 1
1420 CONTINUE
IF(LQ .GT. LKMAXQ(KTH,1), LKMAXQ(KTH,1) = LQ
IF(LKLANS(KTH) .NE. 1) GO TO 1500

***** IF TWO LANES, CHECK THE SECOND LANE FOR QUEUE.

IB1 = IB2 + 1
IB2 = IB2 + LKLNG(KTH)
LQ = 0
DO 1470 IR=IB2,IB1,-1
JTH = BLKARY(IB)
IF(JTH .EQ. 0) GO TO 1470
IF(LQ .NE. 0) GO TO 1460
IF(UASP(JTH) .NE. 0) GO TO 1470
1460 LQ = LQ + 1
1470 CONTINUE
IF(LQ .GT. LKMAXQ(KTH,2)) LKMAXQ(KTH,2) = LQ
1500 CONTINUE
1999 CONTINUE
2000 CONTINUE

C**** WRITE(6,1) CLOCK,(SIGSTA(I1*I2*I3),I2=1,2),I1=1,NUMSIG)
1 FORMAT(///,' CLOCK ',I9,1 SIGNALS:',I9,2I2))
RETURN
END
FUNCTION SPDDIS(I1SEED)
DIMENSION ARY(4)
DATA ARY / 0.07, 0.97, 1.0, /
X=RAN1(I1SEED)
DO 100 I=2,4
IF(ARY(I) .GE. X) GO TO 200
100 CONTINUE
200 SPDDIS = I
RETURN
END

SUBROUTINE STAT(K)

***** THIS SUBROUTINE CLEARS THE STATISTICS AFTER WAR,UP
AND THEN PRINTS THEM AFTER THE FINISH.
C

INCLUDE DEFINS

C

..... IF FINISH* GO TO 1000
IF(K = EQ. 2) GO TO 1000
DO 200 KTH = 1*NUMLKS
DO 100 I = 1*2
LKMAXO(KTH,I) = 0
LKSTOP(KTH,I) = 0
LKDEL(KTH,I) = 0
100 LKVL(KTH,I) = 0
DO 200 I = 1*3
LKDEL(KTH,I) = 0
200 LKVD(KTH,I) = 0
RETURN

C

..... PRINT THE STATISTICS IN THIS SECTION.

C

1000 TOTIM = FINISH - WARMUP
WRITE(6,1)
1 FORMAT(1H1•21X•'VEHICLE COUNT',15X•'VEHICLE VOLUME',21X•
1 'VEHICLE DELAY',14X•'DELAY PFR VEHICLE',/'• LINK LANES',
3 12X•'LANE 1 LANE 2 TOTAL',
1 LANE 1 LANE 2 TOTAL',

DO 1100 KTH = 1*NUMLKS
LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
LKVL1 = LKVL(KTH,1) + 3600 / TOTIM
LKVL2 = LKVL(KTH,2) + 3600 / TOTIM
LKVL = LKVL1 + LKVL2
LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
DELVL = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))
1100 WRITE(6,2) KTH•LKANS(KTH)•LKVL(KTH,1)•LKVL(KTH,2)•LKVTOT,
1 LKVL1•LKVL2•LKVL•LKDELT•DELVL1•DELVL2•DELVL
2 FORMAT(I4•4X•I2•2(1X•3I9)•9X•3I9•2X•3F9.2)
WRITE(6,3)
3 FORMAT(/•14X•'VEHICLE STOPS',11X•'MAX QUEUES',14X•'TURNS',
1 19X•'TURN DFLAY',14X•'TURN DELAY PFR VEHICLE',/•
2 ' LINK LANE 1 LANE 2 TOTAL LANE 1 LANE 2 LEFT',
3 ' STRAIGHT RIGHT LEFT STRAIGHT RIGHT LEFT',
3 ' STRAIGHT RIGHT',/)
DO 1200 KTH = 1*NUMLKS
LKSOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
DELVL1 = LKDEL(KTH,1) / LKVD(KTH,1)
DELVL2 = LKDEL(KTH,2) / LKVD(KTH,2)
DELVL3 = LKDEL(KTH,3) / LKVD(KTH,3)
1200 WRITE(6,4) KTH•LKSOT(KTH,1)•LKSOT(KTH,2)•LKSOT(KTH,3)•LKMANO(KTH,1),
1 LKMAXO(KTH,2)•(LKVD(KTH,1),I = 1*3)•(LKDEL(KTH,1),I = 1*3),
2 DELVL1•DELVL2•DELVL3
4 FORMAT(I4•3(4X•I5)•6X•12•7X•I2•2X•3(3X•I5)•2(4X•I6)•3X•I6,
1 2X•3(1X•F8.2))
5 FORMAT(/•23H INTERSECTION ID NUMBER',14•'12H INPUT LINKS',6X•
1 14H VEHICLE VOLUME',10X•'MAX QUEUE',11X•'13H VEHICLE DFLAY',12X•
2 17H DELAY PFR VEHICLE',14X•'22H LANE 1 LANE 2 TOTAL',4X•
3 14H LANE 1 LANE 2 (4X•'22H LANE 1 LANE 2 TOTAL)'/)
6 FORMAT(17,4X,21H,19,21B,3X,21A,19,2X,2F8.2,F9.2)
7 FORMAT(30X,6H---,6X,6H---,21H TOTAL INTERSECTION
1 BVOLUME =,17,23X,40HAVERAGE INTERSECTION DELAY PFR VEHICLE =,2 F7.2)
8 FORMAT(///,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
9 FORMAT(29H INTERSECTION OUTPUT SUMMARY.)

WRITE(6,9)
NTOT = 0
NDTOT = 0
DO 1275 NTH=1,NUMNOS
   IF(NOTYPE(NTH) .LT. 3) GO TO 1275
   NINT = 0
   NDEL = 0
   WRITE(6,5) NOIDNO(NTH)
   ITH = NOSIG(NTH).
   DO 1250 II=1,4
      KTH = SIGINP(IETHII)
      LKVTOT = LKV(LKTH1) + LKV(LKTH2)
      NINT = NINT + LKV(TOT)
      LKDELT = LKDEL(LKTH1) + LKDEL(LKTH2)
      NDEL = NDEL + LKDFLT
      DELVL1 = LKDEL(LKTH1) / FLOAT(LKV(LKTH1))
      DELVL2 = LKDEL(LKTH2) / FLOAT(LKV(LKTH2))
      DELVLT = LKDELT / FLOAT(LKV(TOT))
   1250 WRITE(6,6) KTH,LKV(LKTH1),LKV(LKTH2),LKVTOT,LKMAXO(KTH1)
   1 LKVMAX(LKTH2),LKDEL(LKTH1),LKDEL(LKTH2),LKNELT,
   2 DELVL1,DELVL2,DELVLT
   NTOT = NTOT + NINT
   NDTOT = NDTOT + NDEL
   DEL = NDEL / FLOAT(NINT)
   WRITE(6,7) NINT,DFL
1275 CONTINUE
   DEL = NDTOT / FLOAT(NTOT)
   WRITE(6,8) DEL
   RETURN
END

SUBROUTINE STATA(I)

C PERIOD BY PERIOD OUTPUT GENERATOR. LKSOUT IS AN ARRAY
C TELLING THE DESIRED OUTPUT LINKS.
C
INCLUDE DEFINES
DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
1 DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
 DATA LKSOUT / 1,5,0,0 /
 PARAMETER N=2
 GO TO (10,20),I
10 CONTINUE
C**** WRITE(6,4)
DO 15 II=1,N
   L1(II)=0
   DEL1(II)=0
   LL1(II)=0
   LS1(II)=0
   LR1(II)=0
15 RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
DO 30 II=1*N
MM = LSOUT(II)
L1P = LKVL(MM,1) + LKVL(MM,2)
N1(II) = L1P - L1(II)
L1(II) = L1P
NL(II) = LKVD(MM,1) - L1(II)
L1(II) = LKVD(MM,1)
NL(II) = LKVD(MM,2) - L1(II)
L1(II) = LKVD(MM,2)
NL(II) = LKVD(MM,3) - L1(II)
L1(II) = LKVD(MM,3)
IDELP = LKDEL(MM,1) * LKDEL(MM,2)
DELAY(II) = DELP - DEL(II)
DEL(II) = DELP
AVG1(II) = DELY(II)/N1(II)
X(II) = IDELP/FL0AT(L1P)
C**** WRITE(6+2) L1P(LKVD(MM,J),J=1,3)
C**** IDELP*FL0AT(L1P)
30 CONTINUE
PUNCH 3*clock, numveh, (N1(II), AVG1(II), X(II)) , II=1*N
RETURN
1 FORMAT(/9H clock = ,17*13H numveh = ,*17)
2 FORMAT(10X,16*I5,13*I8,F10.2,11*I10,315,F8.2)
3 FORMAT(17*I15+4(I5,2F6.1))
4 FORMAT(1H1+12X,10*CUMULATIVE,18X,7Hcurrent,7X,11Hlast period,12X
  1 13X,2SHVOL LF ST RT DFLAY,7X,3HMOE,7X,
  2 2SHVOL LF ST RT DEL/V)
END

FUNCTION TBAGFN(NTH)
C
C THIS FUNCTION GENERATES THE TIMF BETWEEN ARRIVALS AT THE NTH NODE. AN EXPONENTIAL TIME BETWEEN ARRIVALS IS ASSUMED.
C
INCLUDE DEFINS
TBAGEN = -ALOG(RN1(NOSEE(NTH))) * NPARS(NTH+1)
RETURN
END

SUBROUTINE TERMIN(JTH,IB)
C
C THIS SUBROUTINE TERMINATE VEHICLES THAT ARE LEAVING THE NETWORK.
C
INCLUDE DEFINS
C
C FIRST SET BLOCK ON LINK.
BLKARY(IB) = 0
C
C THEN GET COUNT FOR VOLUMF CHECK.
LANE = VLAN(JTH)
KTH = VLK(JTH)
LKVL(KTH,LANE) = LKVL(KTH,LANF) + 1
C
C NEXT ZERO DESIRED SPEED AND LINK NUMBER.
VDSP(JTH) = 0
VLK(JTH) = 0
VHOLD(JTH) = 0
VLAN(JTH) = 0
VTURN(JTH) = 0

C

..... FINALLY DECREASE NUMBER OF VEHICLES IN THE NETWORK.
C
NUMVEH = NUMVEH - 1
C

*** WRITE(6,1) JTH,IB
1 FORMAT('VEHICLE',I4,' TERMINATED FROM BLOCK',I4)
RETURN
END
SUBROUTINE UPDATEV(JTH,NOBLKM,NUM)
C

..... THIS SUBROUTINE UPDATES INFORMATION ON ARRAY FOo
C EACH VEHICLE. CAN BE USED TO GATHER STATISTICS
C IF DESIRED.
C
INCLUDE DEFINS
VMOVE(JTH)=1
C

..... DID VEHICLE DECELERATE, STAY AT SAME SPEED OR
C ACCELERATE THIS TIME PERIOD.
C
IF(VASP(JTH)-NOBLKM) 10,20,30
C

10 VSTATE(JTH)=1
VASP(JTH)=NOBLKM
RETURN
C

..... SAME SPEED.
C
20 VSTATE(JTH)=3
IF(NOBLKM .EQ. 0 .AND. IDUM .NE. 1) VSTATE(JTH)=0
RETURN
C

..... DECELERATION SECTION.
C
30 VSTATE(JTH)=2
VASP(JTH)=NOBLKM
IF(NOBLKM .NE. 0 .OR. IDUM .EQ. 1) RETURN
C

..... VEHICLE HAS DECELERATED TO A STOP.
C
VSTATE(JTH) = 0
KTH = VLK(JTH)
LANE = VLAN(JTH)
LKSTOP(KTH*LANE) = LKSTOP(KTH*LANE) + 1
RETURN
END
SUBROUTINE VCLEAR
C

..... THIS SUBROUTINE SETS MOVF FLAG FOR ALL VEHICLES.
SUBROUTINE VEGEN

C
C THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE
C NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER
C VEHICLE.
C
INCLUDE DEFINS

C**** WRITE(6,2)
2 FORMAT(• VEGEN')
DO 2000 NTH=1;NUMNOS
IF(NOTYPE(NTH) .GT. 1) GO TO 2000

C
C CHECK TO SEE IF READY FOR ANOTHER VEHICLE
C
1000 IF (CLOCK .LT. NOCLK(NTH)) GO TO 2000

C
C FIND OUT IF HOLDING AREA IS AVAILABLE.
C
IBNEW = IHOLD(NOUTP(NTH))
IF(IBNEW .EQ. 0) GO TO 2000

C
C HOLDING AREA READY, FIND AN AVAILABLE VEHICLE.
C
1105 CONTINUE
DO 1110 JTH=1;MAXVEH
IF(VDSP(JTH) .EQ. 0) GO TO 1110
1110 CONTINUE
CALL DUMP(0 • VEGEN • MAXVEH • '7)

C
C NOW FILL IN HIS PARAMETERS.
C
1121 VDSP(JTH) = SPDDIS(ISFED)
VASP(JTH) = VDSP(JTH)
VSTATE(JTH) = 3
NUMVEH = NUMVEH + 1

C
C PLACE VEHICLE IN HOLDING AREA
C
C**** WRITE(6*1) JTH+1
1 FORMAT(• NEW VEHICLE' NUMBER *•I4•' VDSP = *•I2)
CALL HOLDIN(JTH,IBNEW,0,0)

C
C COMPUTE TIME OF NEXT ARRIVAL
C
NOCLK(NTH) = NOCLK(NTH) + TBAGFN(NTH)

C
C READY FOR ANOTHER.
C
GO TO 1000

2000 CONTINUE
RETURN
END
APPENDIX D

FORTRAN LISTING OF ZONE MODEL
DEFINS PROCEDURE

...... SETUP FOR NODES.

PARAMETER MAXNO=35
REAL NOPARS,NOCCLK
COMMON /NODES/ NOTDNO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO,4),
1 NOSEED(MAXNO),NOPARS(MAXNO,2),NOOUTP(MAXNO),
2 NOSIG(MAXNO),NOCLK(MAXNO),NUMNOS

...... SETUP FOR SIGNALS.

PARAMETER MAXSIG=20
INTEGER AMBERT,SIGNP,SIGCYC,SIGSTA,SIGOFF,SIGGRN,SIGRED,SIGCLK
COMMON /SIG/ SIGCYC(MAXSIG),SIGSTA(MAXSIG,2),SIGOFF(MAXSIG,2),
1 SIGGRN(MAXSIG,2),SIGRED(MAXSIG,2),SIGCLK(MAXSIG,2),
2 SIGNP(MAXSIG,4),NUMSIG,AMBERT

...... SETUP FOR LINKS.

PARAMETER MAXLKS=100
REAL LKPROB
COMMON /LINKS/ LKTD(MAXLKS*2),LKLNG(MAXLKS),LKLANS(MAXLKS),
1 LKFSTD(MAXLKS),LKNON(MAXLKS),LKNYP(MAXLKS),LKOPOS(MAXLKS),
2 LKARM(MAXLKS),LKLEFT(MAXLKS),LKPROB(MAXLKS,2),LKNEST(MAXLKS,3),
3 NUMLKS

...... SETUP FOR ZONES.

PARAMETER MAXBLK=2000
COMMON /BLOCKS/ BLKARY(MAXBLK),LENGTH,NBLKS
INTEGER BLKARY

...... GENERAL COMMON VARIABLES.

INTEGER CLOCK,DELT,FINISH,CAP,WARMUP
COMMON CLOCK,ISEED,LIMCH,DELT,FINISH,CAP,WARMUP,NUMVEH,ISPEED

...... STAT SETUP.

COMMON /STAT/ LKDEL(MAXLKS*2),LKMAXQ(MAXLKS,2),LKVL(MAXLKS*2)

* * * * * * * * * * * * * * * * * * * *
* DEFINITION OF VARIABLES USED IN THE PROGRAM. *
* * * * * * * * * * * * * * * * * * * *

AMBERT - AMBER TIME IN SECONDS.
BLKARY(IB) - ARRAY OF BLOCKS.
CAP - THE CAPACITY OF A ZONE.
CLOCK - MASTER SIMULATION CLOCK (IN SECONDS).
DELT - THE TIME STEP (IN SECONDS).
FINISH - THE CUTOFF TIME (IN SECONDS).
ITH - INDEX USED TO POINT TO SPECIFIC SIGNAL.
**ISEED** - SEED TO THE RANDOM NUMBER GENERATOR.

**ISPEED** - AVERAGE SPEED IN NETWORK (FEET PER SECOND).

**KTH** - INDEX USED TO POINT TO SPECIFIC LINK.

**LENGTH** - LENGTH OF A ZONE (IN FEET).

**LKARM(KTH)** - 1 IF MAJOR APPROACH, 2 IF MINOR.

**LKDDEL(KTH)** - CUMULATIVE LINK DELAY BY LANES.

**LKDEST(KTH)** - LINK DESTINATIONS: LEFT, STR, RIGHT.

**LKFSTB(KTH)** - LINK'S FIRST BLOCK.

**LKID(KTH)** - SYMBOLIC I.D. OF TAIL(1) AND HEAD(2).

**LKLANS(KTH)** - NUMBER OF LANES.

**LKLLEFT(KTH)** - FLAG INDICATING A VEHICLE DELAYED MAKING A LEFT TURN.

**LKLNG(KTH)** - LINK LENGTH (IN BLOCKS).

**LKMAXQ(KTH)** - MAXIMUM QUEUE LENGTH FOUND BY LANES.

**LKNOD(KTH)** - INDEX TO NODE AT HEAD OF LINK.

**LKOPOS(KTH)** - LINK APPPOSING A LEFT TURN.

**LKPROB(KTH)** - CUMULATIVE PROB OF LEFT & STR. MOVE.

**MAXBLK** - MAXIMUM POSSIBLE NUMBER OF BLOCKS.

**MAXLKS** - MAXIMUM POSSIBLE NUMBER OF LANES.

**MAXNO** - MAXIMUM POSSIBLE NUMBER OF NODES.

**MAXSIG** - MAXIMUM POSSIBLE NUMBER OF SIGNALS.

**NBLKS** - NUMBER OF BLOCKS IN MODFL.

**NOCLK(NTH)** - TIME OF NEXT ARRIVAL AT NTH NODE.

**NOINNO(NTH)** - SYMBOLIC NODE IDENTIFIER.

**NONXT(NTH)** - SYMBOLIC IDENTIFIERS OF ADJ. NODES.

**NOOUTP(NTH)** - INDEX TO LINKS TAKING FROM THIS NODE.

**NODPARS(NTH)** - PARAMETERS FOR GENERATE NODES.

**NOSIG(NTH)** - INDEX TO SIGNAL FOR NTH NODE.

**NOTYPE(NTH)** - NODE TYPE:

0 = BOTH GENERATE AND TERMINATE
1 = GENERATE ONLY
2 = TERMINATE ONLY
4 = TWO WAY STOP
5 = FIXED TIMED CONTROLLED

**NTH** - INDEX USED TO POINT TO SPECIFIC NODE.

**NUMLKS** - NUMBER OF LINKS IN MODFL.

**NUMNOS** - NUMBER OF NODES.

**NUMSIG** - NUMBER OF SIGNALS.

**NUMVEH** - NUMBER OF VEHICLES IN NETWORK.

**SIGCLK(I)TH** - SIGNAL CLOCK.

**SIGCYC(I)TH** - CYCLE LENGTH OF ITH SIGNAL.

**SIGGRN(I)TH** - GREEN TIME.

**SIGINP(I)TH** - LINKS POINTING AT SIGNAL.

**SIGOFF(I)TH** - OFFSET FOR SIGNAL.

**SIGRED(I)TH** - RED TIME.

**SIGSTA(I)TH** - STATE OF THE SIGNAL.

**WARMUP** - TIME SPENT BEFORE GATHERING STATISTICS.
C
C END
C
***** THIS IS THE MAIN PROGRAM FOR THE ZONE MODEL.
THE CLOCK IS IN UNITS OF SECONDS.
C
INCLUDE DEFINES.LIST
C
***** INITIALIZATION
NTH = 0
NBLKS = 1
ISPEED = 53
C
***** RUN PARAMETERS
C
READ(5,1) IRUN,INFT,AMBERT,ISFFD,DELT,FINISH,WARMUP
1 FORMAT( )
MODEL = 'ZONE'
WRITE(22) MODEL,IRUN,INET,ISEED
PUNCH 6,IRUN,INET,ISEED
6 FORMAT('ZONE MODEL IRUN = *14,' INFT = *14,' ISEED = *112)
LIMDCH = DELT / 2
IF(LIMDCH .LE. 0) LIMDCH = 1
WARMUP = WARMUP * 60
FINISH = FINISH * 60 + WARMUP
CLOCK = -DELT
LENGTH = ISPEED * DELT
CAP = LENGTH / 20.0 * 0.5
WRITE(6,3)
3 FORMAT('1*************************** ZONE MODEL'
1 '***************************')
WRITE(6,2) IRUN,INET,ISEED,DELT,FINISH,WARMUP,ISPEED,LENGTH,CAP
2 FORMAT( ' RUN NUMBER '15,' NETWORK '15,' ISEED '
1 19//' DELT '18//' FINISH '18//' WARMUP '
2 18//' ISPEED '13',' ZONE LENGTH '14',' ZONE CAPACITY '
3 13//' )
C
***** CALL INPUT SUBROUTINES.
C
CALL EXTERN(NTH)
CALL INTERN(NTH)
CALL INPLKS
C
***** CALL SETUP TO FINISH DATA MANIPULATION.
C
CALL SETUP
CALL DUMP(0,'DATA ','CHECK ',0)
C
***** THE SIMULATION.
C
1000 CLOCK = CLOCK + DELT
IF(MOD(CLOCK,90).EQ.0) CALL STATA(2)
WRITE(22) CLOCK,NIMVEH
IF(CLOCK .GE. WARMUP) GO TO 2000
CALL SIGCHK
CALL VEGEN
CALL INTERL
CALL INTRAL
CALL HOLDOU
GO TO 1000

***** WARMUP OVER, CLEAR STATISTICS.

2000 CALL STAT(1)
CALL STATA(1)
GO TO 2300

2100 CLOCK = CLOCK + DFLT
IF(MOD(CLOCK+90),90) .EQ. 0 CALL STATA(2)
WRITE(22) CLOCK nuruVEH
IF(CLOCK .GE. FINISH) GO TO 3000

2300 CALL SIGCHK
CALL VEGEN
CALL INTERL
CALL INTRAL
CALL HOLDOU
GO TO 2100

***** SIMULATION OVER, PRINT STATISTICS.

3000 CALL STAT(2)
ENDFILE 22
STOP
END

SUBROUTINE DUMP(M,N1,N2,K)

***** THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES
IN CASE OF AN ERROR. THE VALUE OF K DETERMINES
THE TYPE OF DUMP:
0 - SFT UP CHKCK, CALL RETURN
AFTER PRINTOUT ON NODES, eTC.
1 - ERROR ON INPUT,
> 1 - ERROR WHILE RUNNING - LINF NUMBER.

INCLUDE DEFINS

***** GENERAL INFORMATION DUMP.
WRITE(6,1) K
1 FORMAT(///, ERROR DUMP OUTPUT. K = ' 'I5')
IF(K .GE. 2) GO TO 100
WRITE(6,2) M,N1,N2
2 FORMAT(CODE ' 'I3', MESSAGE - ' '2A6)
GO TO 1001
100 WRITE(6,3) M,N1,N2
3 FORMAT( LINK OR SIG ' 'I5', NOW BEING PROCESSED.' '/., MESSAGE -
1 'IX' '2A6)

***** NODE OUTPUT.

1001 MAX = MAXNO
WRITE(6,1010) MAX,NUMNOS
1010 FORMAT(///, NODE OUTPUT MAX = ' '2,12X,
1 'NUMBER = ' '12//', 'INDEX ID NEXT NODE ID',
2 ' TYPE GEOM INDEX LINKS IN LINK OUT SIGNAL',
3 ' FSTB CLOCK PARAMETERS'//)
DO 101 I=1,NUMSOS
101 WRITE(6,111) I,NOIDNO(I),NONTXT(I*K),K=1,4,NOTYPC(I),
1 NOOUTP(I),NOSIG(I)*NOCLK(I),
2 (NOPAR(I,K),K=1,2)
1011 FORMAT(14,6X,12,3X,413,6X,12,5X,2X,2X,16X,7X,12,8X,12,5X,
1 3X,3X,F5.0,2X,F5.0,1X,F5.0)

C ****** SIGNAL OUTPUT

C MAX = MAXSIG
LAM = AMBERT
WRITE(6,1020) MAX,NUMSIG,LAM
1020 FORMAT(///,SIGNAL OUTPUT MAX = 'I2,' NUMBER = 'I3,/
1 ' INDEX CYCLE',2(6X,OFFSET GREEN RED STATE CLOCK NXT '/,
2 'AMBER TIME = 'I2,///,23X,** ** MAJOR**, 
3 'INDEX CYCLE',2(6X,OFFSET GREEN RED STATE CLOCK NXT '/,
4 )
DO 102 I=1,NUMSIG
102 WRITE(6,1021) I,STGCYC(I),SIGOFF(I,K),SIGGRN(I,K),SIGRED(I,K),
1 SIGSTA(I,K),SIGCLK(I,K),K=1,2
1021 FORMAT(I4,5X,I3,2(BX,I3,5X,I3,5X,I2,3X,I5,4X,2X))

C ****** LINK OUTPUT

C MAX = MAXLKS
WRITE(6,1030) MAX,NUMLKS
1030 FORMAT(///,LINK OUTPUT MAX = 'I3,' NUMBER = 'I3,/
1 'INDEX ID NO. LENGTH LANES NODE LK OPOS LK**
2 'DESTINATIONS PROBABILITIES FSTB ARM LEFT**/
DO 103 I=1,NUMLKS
103 WRITE(6,1031) I,LKID(I),LKIN(I,2),LKLNG(I),LKLANS(I),LKNOD(I),
1 LKOPOS(I),LKDEST(I,K),K=1,3,(LKRPR(I,K),K=1,2),LKFRSTR(I),
2 LKARM(I),LKLEFT(I)
1031 FORMAT(I5,4X,13,7X,2X,16X,12,7X,I3,2I6,3X,I2,2X,
1 2F7.3,5X,I4,3X,I1,5X) IF(K.EQ.0) RETURN
IF(K.EQ.1) STOP

C ****** BLOCK OUTPUT

C MAX = MAXRLK
WRITE(6,1040) MAX,NBLKS
1040 FORMAT(*1BLK OUTPUT MAX = 'I4,' NUMBER = 'I4,/
1 'FIRST * VALUE .....................................')
1 I1 = -24
104 I1 = I1 + 25
I2 = I1 + 24
WRITE(6,1041) I1,(BLKARY(I),I1=I1,I2)
1041 FORMAT(I5,*,',I5)
IF(I2 .LT. NBLKS) GO TO 104
STOP
END
SUBROUTINE EXTERN(NTH)

C THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
C NODES ONLY. CONTROL IS RETURNED TO THE CALLING
C PROGRAM AFTER A BLANK CARD IS FOUND
THIS SUBROUTINE TAKES VEHICLES OUT OF A HOLDING AREA AND MOVES THEM ONTO THE LINK.

INCLUDE DEFINS

**** WRITE(6,1)
1 FORMAT(///, 'HOLDOU SUBROUTINE')
DO 100 KTH=1, NUMLKS

IB = LKFSTB(KTH) + LKLNG(KTH) * LKLANS(KTH)
IF(LKLANS(KTH) .EQ. 2) GO TO 35

C SINGLE LANE STREET.
IF(BLKARY(IB) .EQ. 0) GO TO 30
IBP = IB - 1
IF(BLKARY(IB) + BLKARY(IBP) - CAP) 10,20,30

C NOT FULL.
LKVL(KTH) = LKVL(KTH) + BLKARY(IB)
BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
BLKARY(IB) = 0
C WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
C 2 FORMAT( 'LINK #',I4,' BLK #',I4,' CONT #',I4,' BLK #I',1*CONT#I4)
GO TO 100

C FULL.
LKVL(KTH) = LKVL(KTH) + BLKARY(IB)
BLKARY(IBP) = CAP
C WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
GO TO 100

C OVERFLOW BACK INTO HOLDING AREA.
LKVL(KTH) = LKVL(KTH) + CAP - BLKARY(IBP)
BLKARY(IBP) = BLKARY(IBP) + BLKARY(IBP) - CAP
C WRITE(6,2) KTH,IB,BLKARY(IB),IBP,BLKARY(IBP)
GO TO 100

C TWO LANE STREET, MUST CHECK BOTH AREAS.
C 35 IBP = LKFTB(KTH) + LKLN(KTH) - 1
   IF(BLKARY(IB) .EQ. 0) GO TO 65
   IF(BLKARY(IB)+BLKARY(IBP)-CAP) 40,50,60
C C ***** NOT FULL.
C 40 LKVL(KTH+1) = LKVL(KTH+1) + BLKARY(IB)
   BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
   BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
   GO TO 65
C C ***** FULL.
C 50 LKVL(KTH+1) = LKVL(KTH+1) + BLKARY(IB)
   BLKARY(IBP) = CAP
   BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
C C ***** OVERFLOW BACK INTO HOLDING AREA.
C 60 LKVL(KTH+1) = LKVL(KTH+1) + CAP - BLKARY(IBP)
   BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
   BLKARY(IBP) = CAP
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
C C ***** SECOND HOLDING AREA OF TWO LANE STREET.
C 65 IB = IB + 1
   IF(BLKARY(IB) .EQ. 0) GO TO 100
   IBP = IB - 2
   IF(BLKARY(IB)+BLKARY(IBP)-CAP) 70,80,90
C C ***** NOT FULL.
C 70 LKVL(KTH+2) = LKVL(KTH+2) + BLKARY(IB)
   BLKARY(IBP) = BLKARY(IBP) + BLKARY(IB)
   BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
   GO TO 100
C C ***** FULL.
C 80 LKVL(KTH+2) = LKVL(KTH+2) + BLKARY(IB)
   BLKARY(IBP) = CAP
   BLKARY(IB) = 0
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
   GO TO 100
C C ***** OVERFLOW BACK INTO HOLDING AREA.
C 90 LKVL(KTH+2) = LKVL(KTH+2) + CAP - BLKARY(IBP)
   BLKARY(IB) = BLKARY(IB) + BLKARY(IBP) - CAP
   BLKARY(IBP) = CAP
C**** WRITE(6,2) KTH,IB,BLKARY(IB),TRP,BLKARY(IBP)
100 CONTINUE
RETURN
END
FUNCTION IBHOLD(KTH)
C C ***** THIS FUNCTION RETURNS A N IF NO HOLDING
SPACE IS AVAILABLE. OTHERWISE IT RETURNS THE
INDEX TO THE BLOCK ARRAY.

INCLUDE DEFINS

IBHOLD = 0
IB = LKFSTB(KTH) + LKLNG(KTH) * LKLANS(KTH)
IF(LKLANS(KTH) *EQ. 2) GO TO 10

***** SINGLE LANE STREET.

IF(BLKARY(IB) .GE. CAP) RETURN
5 IBHOLD = IB
RETURN

***** TWO LANE STREET.

10 IF(RN1(ISEED) .LT. 0.5) GO TO 20
IF(BLKARY(IB) .LT. CAP) GO TO 5
IB = IB + 1
IF(BLKARY(IB) .LT. CAP) GO TO 5
RETURN

20 IB = IB + 1
IF(BLKARY(IB) .LT. CAP) GO TO 5
IB = IB - 1
IF(BLKARY(IB) .LT. CAP) GO TO 5
RETURN

FUNCTION IGAP(KTH)

THIS FUNCTION TESTS THE GAP THE VEHICLE IS ORSING.
A 0 MEANS GO AND A 1 MEANS STOP.

INCLUDE DEFINS
DIMENSION PROR(20)
DATA PROB / -.02, -.1, 0.92, 0.22, 0.36, 0.5, 0.61, 0.74, 1, .81, .84, .9, .92, .95, .96, .97, .98, .99, .995, .9995 /
L = LKLNG(KTH)
IF(LKLANS(KTH) *EQ. 2) GO TO 20

***** CHECK SINGLE LANE.

IF(LKLEFT(KTH) *EQ. 1) GO TO 70
IB2 = LKFSTB(KTH) - 1
5 DO 10 I=1,L
IB2 = IB2 + 1
IF(BLKARY(IB2) *NE. 0) GO TO 50
10 CONTINUE
GO TO 70

***** CHECK TWO LANE STREETS.

20 IB1 = LKFSTB(KTH) - 1
IB2 = IB1 + L
IF(LKLEFT(KTH) *EQ. 1) GO TO 5
DO 30 I=1,L
IB1 = IB1 + 1
30 CONTINUE
IB2 = IB2 + 1
IF(BLKARY(IBl) .NE. 0 .OR. BLKARY(IR2) .NE. 0) GO TO 50
30 CONTINUE
GO TO 70
50 GAP = (I-1) * DElt
IF(GAP .GE. 20.0) GO TO 70
IF(GAP .LE. 2) GO TO 60
INDEX = GAP + 0.5
IF(RNL(ISEED) .LE. PROB(INDEX)) GO TO 70
C
C     *** REJECT.***
60 IGAP = 1
RETURN
C
C     *** ACCEPT.***
70 IGAP = 0
RETURN
END
SUBROUTINE INPLKS
C
C THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
C AFTER ALL DATA CARDS HAVE BEEN READ, THE ROUTINE
C CALCULATES SOME CROSS REFERENCE INDICES.
C
   INCLUDE DEFS
C
C THIS IS THE LOCAL DECLARATION.
C
INTEGER ALPHA
KTH=0
1000 KTH=KTH+1
READ(5,101) ALPHA,LKID(KTH,1),LKID(KTH,2),LKLNG(KTH),LKLANS(KTH),
1 LKOPOS(KTH),LKPORB(KTH,1),LKPORB(KTH,2),(LKDEST(KTH,K),K=1,3)
C
C     *** IF IT IS A PLANK CARD, GO TO 20n0***
IF(ALPHA .EQ. 1H ) GO TO 20n0
C
C     *** CALCULATE THE BLOCKS PER LANE AND THE NUMBER OF
C     BLOCKS USED THUS FAR.*
LKLNG(KTH)=LKLNG(KTH)/FLOAT(LENGTH) + 0.5
IF(LKLNG(KTH) .LT. 1) LKLNG(KTH) = 1
LKSTB(KTH)=NRLKS
NBLKS=NBLKS+LKLNG(KTH)*LKLANS(KTH)+LKLANS(KTH)
C
C     *** FIND NODE NUMBER OF THE HEAD NODE.***
DO 1015 NTH=1,NUMNOS
IF(NODEO(NTH) .EQ. LKID(KTH,2)) GO TO 1015
1010 CONTINUE
CALL DUMP(KTH,'INPLKS','DO1015
1015 LKNOD(KTH)=NTH
DO 1016 I=1,4
1016 CONTINUE
CALL DUMP(KTH,'INPLKS','DO1016
1017 LKARM(KTH) = 2 - MOD(I,2)
C       SET UP THE TURNING PROBABILITIES.

1060 LKPROB(KTH+2)=LKPROB(KTH+1)*LKPROB(KTH+2)
   IF(LKPROB(KTH,2) .GE. 0.9985) LKPROB(KTH+2) = 1.0
   GO TO 1000

C       FIND LINK OPPOSED.

2000 NUMLKS=KTH-1
   IF(NUMLKS .GE. MAXLKS) CALL NUMP(NUMLKS, INPLKS, MAXLKS)
   DO 2500 KTH=1, NUMLKS
   IF(LKOPOS(KTH,1) .EQ. 0) GO TO 2500
   DO 2400 KTH2=1, NUMLKS
   IF(LKID(KTH2,2) .NE. LKID(KTH+2)) GO TO 2400
   IF(LKOPOS(KTH) .EQ. LKID(KTH2,1)) GO TO 2450
   2400 CONTINUE
   CALL DUMP(KTH, INPLKS, LKOPOS, 1)
   2450 LKOPOS(KTH)=KTH2
   2500 CONTINUE

RETURN

101 FORMAT(A1,2I3,I4,I1,I3,2F3.3,3I3)
END

SUBROUTINE INTERL

C       THIS SUBROUTINE MOVES VEHICLES FROM ONE LINK TO
C       THE NEXT.

C       INCLUDE DEFINITIONS

C**** WRITE(6,5)
   DO 200 KTH=1, NUMLKS
C       SET UP INITIAL VARIABLES.
   LANE = LKLANS(KTH)
   NTH = LKNOD(KTH)
   IR = LKFSTB(KTH)
   IF(LANE .NE. 1) GO TO 6
C       CHECK FOR CARS TO MOVE.
   IF(BLKARY(IB) .EQ. 0) GO TO 200
   IF(LKID(IB,1) .EQ. 0.AND. BLKARY(IB2) .EQ. 0) GO TO 200
C       TERMINATE SECTION.
   6 IB2 = IB + LKLNG(KTH)
   IF(LKARY(IB2) .EQ. 0) GO TO 200
   10 NUMVEH = NUMVEH - BLKARY(IB)
   BLKARY(IB) = 0
C**** WRITE(6,5) KTH, IB
   IF(LANE .EQ. 1) GO TO 200
   NUMVEH = NUMVEH - BLKARY(IB2)
   BLKARY(IB2) = 0
WRITE(6,3) KTH, IB
GO TO 200

***** CHECK SIGNAL STATE FIRST.

15 ITH = NOSIG(NTH)
I = LKARM(KTH)
IF(SIGSTA(ITH, I) .GT. 20, 25

***** AMBFR LIGHT WILL PERMIT ONE LEFT TURN IF WAITING.

20 IF(LKLEFT(KTH) .GT. 0) GO TO 30

***** STOPPING VEHICLES.

25 LKDEL(KTH, 1) = LKDEL(KTH, 1) + RLKARY(1) * DELT

***** WRITE(6,4) KTH, IB

GO TO 200

30 KTHCHK = LKDEST(KTH, 1)
IBP = IBHOLD(KTHCHK)
IF(IBP .EQ. 0) GO TO 25
LKLEFT(KTH) = 0
BLKARY(IBP) = BLKARY(IBP) + 1
BLKARY(IB) = RLKARY(IR) - 1
LKDEL(KTH, 1) = LKDEL(KTH, 1) + RLKARY(1B) * DELT

***** WRITE(6,1) KTH, IB, IBP

***** WRITE(6,4) KTH, IB

GO TO 200

35 KTHCHK = LKDEST(KTH, 1)
IBP = IBHOLD(KTHCHK)
IF(IBP .EQ. 0) GO TO 25
LKLEFT(KTH) = 0
BLKARY(IBP) = BLKARY(IBP) + 1
BLKARY(IB) = RLKARY(IR) - 1
LKDEL(KTH, 2) = LKDEL(KTH, 2) + RLKARY(1B2) * DELT

***** WRITE(6,4) KTH, IB2

GO TO 200

50 NOMOVD = 0
IF(LANE .EQ. 2) GO TO 100

***** CHECK THE TURN FLAG.

51 IF(BLKARY(IB) .EQ. 0) GO TO 200
IF(NOMOVD .GE. LIMDCH) GO TO 65

***** PICK A DESTINATION FOR THE VEHICLE.

X = RNKISEED
IF(X .LE. LKPROB(KTH, 1)) GO TO 60
IF(X .LE. LKPROB(KTH, 2)) GO TO 55

***** RIGHT.
KTHCHK = LKDEST(KTH+3)
GO TO 70
C
...... STRAIGHT.
55 KTHCHK = LKDEST(KTH+2)
GO TO 70
C
...... LEFT.
60 KTHCHK = LKDEST(KTH+1)
IBP = IBHOLD(KTHCHK)
IF(IGAP(LKOPOS(KTH)) .NE. 1 .AND. IBP .NE. 0) GO TO 5
C
...... LEFT TURN HANGUP.
C
LKLEFT(KTH) = 1
65 LKDEL(KTH+1) = LKDEL(KTH+1) + RLKARY(IB) * DELT
C**** WRITE(6,2) KTH,IB,KTHCHK
GO TO 200
C
...... CHECK HOLDING AREA.
C
70 IBP = IBHOLD(KTHCHK)
IF(IBP .EQ. 0) GO TO 65
C
...... MOVE VEHICLE INTO HOLDING AREA.
C
75 LKLEFT(KTH) = 0
BLKARY(IB) = BLKARY(IB) - 1
BLKARY(IBP) = BLKARY(IBP) + 1
NOMOVD = NOMOVD + 1
C**** WRITE(6,1) KTH,IB,KTHCHK
GO TO 51
C
...... TWO LANE STREET, TAKE LANES ONE AT A TIME.
C
100 XTEST = LKPROR(KTH+1)*2
IF(LKLEFT(KTH) .NE. 0) GO TO 105
C
101 IF(BLKARY(IB) .EQ. 0) GO TO 150
IF(NOMOVD .GE. LIMDCH) GO TO 110
C
...... PICK DESTINATION.
C
IF(RNI(ISEED) .GT. XTEST) GO TO 120
C
...... LEFT.
105 KTHCHK = LKDEST(KTH+1)
IBP = IBHOLD(KTHCHK)
IF(IGAP(LKOPOS(KTH)) .NE. 1 .AND. IBP .NE. 0) GO TO 25
C
...... MARK DELAY AND GO TO NEXT LANE.
C
LKLEFT(KTH) = 1
110 LKDEL(KTH+1) = LKDEL(KTH+1) + RLKARY(IR) * DELT
C**** WRITE(6,2) KTH,IB,KTHCHK
GO TO 150
C
...... STRAIGHT.
120 KTHCHK = LKDEST(KTH+2)
   IBP = IBHOLD(KTHCHK)
   IF( IBP .EQ. 0) Go TO 110
   C
   **** MOVE INTO HOLDING AREA.
   C
125 LKLEFT(KTH) = 0
   BLKARY(IBP) = BLKARY(IBP) + 1
   BLKARY(IB) = BLKARY(IB) - 1
   NOMOVD = NOMOVD + 1
   C**** WRITE(6+1) KTH+IBP,KTHCHK
   Go TO 101
   C
   **** NOW CHECK THE SECOND LANE.
   C
150 XTEST = (1.0 - LKPROB(KTH+2)) * 2
   NOMOVD = 0
151 IF(BLKARY(IB2) .EQ. 0) Go TO 200
   IF(NOMOVD .GE. LIMDCH) Go TO 170
   C
   **** PICK A DESTINATION.
   C
   IF(RNKISEED) .LT. XTEST) Go TO 155
   C
   **** STRAIGHT.
   KTHCHK = LKDEST(KTH+2)
   Go TO 160
   C
   **** RIGHT.
155 KTHCHK = LKDEST(KTH+3)
   C
160 IBP = IBHOLD(KTHCHK)
   IF( IBP .EQ. 0) Go TO 170
   BLKARY(IBP) = BLKARY(IBP) + 1
   BLKARY(IB2) = BLKARY(IB2) - 1
   NOMOVD = NOMOVD + 1
   C**** WRITE(6+1) KTH+IBP,KTHCHK
   Go TO 151
170 LKDEL(KTH+2) = LKDEL(KTH+2) + BLKARY(IB2) * DELT
   C**** WRITE(6+2) KTH+IBP,KTHCHK
200 CONTINUE
   1 FORMAT(' LINK ',I4,' MOVE VEH FROM BLK ',I4,' TO LK ',I4)
   2 FORMAT(' LINK ',I4,' VEH IN ',I4,' BLOCKED FOR LINK ',I4)
   3 FORMAT(' LINK ',I4,' TERMINATE VEHICLES IN BLK ',I4)
   4 FORMAT(' LINK ',I4,' VEH IN BLK ',I4,' STOPPED FOR LIGHT. ')
   5 FORMAT//(, ' INTERI SURROUTINE.')
   RETURN
END
SUBROUTINE INTERN(NTH)
   C
   C THIS SUBROUTINE READS DATA CARDS ON INTERIOR
   C NODES ONLY. CONTROL IS RETURNED TO THE CALLING
   C PROGRAM AFTER A BLANK CARD IS FOUND.
   C
   INTEGER ALPHA,TEMP
   INCLUDE DEFINS
   ITH = 0
   C
1000 NTH = NTH + 1
READ(5,101) ALPHA,NODNO(NTH),NOSIG(NTH),NOSIGCYC(NTH),NOSIG(NTH),NOSIG(NTH)
IF(ALPHA .EQ. 'H') GO TO 3000
C
C ****** IF NONSIGNALIZED RETURN TO READING NEXT NODE.
C
IF(NOSIG(NTH) .LT. 4) GO TO 1000
C
C ****** SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
ITH = ITH + 1
NOSIG(NTH) = ITH
READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH),SIGGRN(ITH),
1 SIGOFF(ITH),SIGGRN(ITH)
C
C ****** CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C
DO 2100 I=1,2
SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
TEMP = SIGOFF(ITH,I) + SIGGRN(ITH,I)
IF(TEMP .LT. SIGCYC(ITH)) GO TO 1000
TEMP = TEMP + AMBRFRD
IF(TEMP .LT. SIGCYC(ITH)) GO TO 2005
SIGSTA(ITH,I) = 2
SIGCLK(ITH,I) = SIGOFF(ITH,I)
GO TO 2100
2005 SIGSTA(ITH,I) = 1
SIGCLK(ITH,I) = TFMP - SIGCYC(ITH)
GO TO 2100
2010 SIGSTA(ITH,I) = 0
SIGCLK(ITH,I) = TFMP - SIGCYC(ITH)
GO TO 2100
2100 CONTINUE
GO TO 1000
3000 NUMNOS = NTH - 1
IF(NUMNOS .GE. MAXNO) CALL DUMP(NUMNOS,'INTERN','MAXNO',1)
NUMSIG = ITH
IF(NUMSIG .GE. MAXSIG) CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
RETURN
C
C ****** FORMAT STATEMENTS
C
101 FORMAT(A1,I3,I2,4I3,I1)
102 FORMAT(A1,I3,5I3)
END
SUBROUTINE INTRAL
C
C ****** THIS SUBROUTINE MOVES VEHICLES ON THE LINKS FROM
C ONE ZONE TO THE NEXT.
C
C INCLUDE DEFINES
C
C**** WRITE(6,1)
1 FORMAT('//', INTAR SUBROUTINE')
2 FORMAT(' LINK ',I4,' LANE ',I4,' LIFST ',I4,' IBLST ',I4)
3 FORMAT(' BLOCK ',I4,' CONT ',I4,' BLOCK ',I4,' CONT ',I4)
DO 100 KTH=1,NUMLKS
1000 NTH = NTH + 1
READ(5,101) ALPHA,NOSIG(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4)
IF(ALPHA .EQ. '1H') GO TO 3000
**** IF NONSIGNALIZED RETURN TO READING NEXT NODE.
IF(NOTYPE(NTH) .EQ. 'LF', 4) GO TO 1000
**** SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
ITH = ITH + 1
NOSIG(NTH) = ITH
READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGGRN(ITH,1),
1 SIGOFF(ITH,2),SIGGRN(ITH,2)
**** CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
DO 2100 I=1,2
SIGRED(ITH,I) = SIGCYC(ITH) - SIGGRN(ITH,I) - AMBERT
TEMP = SIGOFF(ITH,I) + SIGGRN(ITH,I)
IF(TEMP .GT. SIGCYC(ITH)) GO TO 2010
TEMP = TEMP + AMBFRNT
IF(TEMP .GT. SIGCYC(ITH)) GO TO 2005
SIGSTA(ITH,I) = 2
SIGCLK(ITH,I) = SIGOFF(ITH,I)
GO TO 2100
2005 SIGSTA(ITH,I) = 1
SIGCLK(ITH,I) = TEMP - SIGCYC(ITH)
GO TO 2100
2010 SIGSTA(ITH,I) = 0
SIGCLK(ITH,I) = TFMP - SIGCYC(ITH)
GO TO 2100
2100 CONTINUE
GO TO 1000
3000 NUMNOS = NTH - 1
IF(NUMNOS .GE. MAXNO) CALL DUMP(NUMNOS,'INTERN','MAXNO',1)
NUMSIG = ITH
IF(NUMSIG .GE. MAXSIG) CALL DUMP(NUMSIG,'INTERN','MAXSIG',1)
RETURN
**** FORMAT STATEMENTS
101 FORMAT(A1,I3,I2,I4,I3,I1)
102 FORMAT(A1,I3,I5,I3)
END
SUBROUTINE INTRAL
**** THIS SUBROUTINE MOVES VEHICLES ON THE LINKS FROM
** ONE ZONE TO THE NEXT.
INCLUDE DEFINS
C**** WRITE(6,1)
1 FORMAT(/' INTRAL SUBROUTINE' )
2 FORMAT(' LINK ',I4,' LANE ',I4,' INFST ',I4,' IBLST ',I4)
3 FORMAT(' BLOCK ',I4,' CONT ',I4,' BLOCK ',I4,' CONT ',I4)
DO 100 KTH=1,NULMLKS
THIS FUNCTION GENERATES UNIFORMLY DISTRIBUTED
RANDOM NUMBERS BETWEEN 0 AND 1

ISEED = ISEED * 18533
IF (ISEED .LT. 0) ISEED = ISEED + 34359738367 + 1
RN1=ISEED*2.0**(-35)
RETURN
END

SUBROUTINE SETUP

..... THIS SUBROUTINE DOES THE FINAL SETUP ON ALL DATA
FILES BEFORE THE SIMULATION BEGINS.

INCLUDE DEFINS

..... FOR EACH NODE
DO 2000 NTH=1,NUMNOS
IF(NOTYPE(NTH) .GT. 2) GO TO 1500
IF(NOTYPE(NTH) .EQ. 2) GO TO 2000

..... GENERATE NODES; SET TIME OF FIRST ARRIVAL.
NOPARS(NTH+1) = 3600. / NOPARS(NTH+1)
DO 1001 I=1,100
1001 A = RN1(ISEED)
NOSEED(NTH) = ISEED
NOCLK(NTH) = TBAGFN(NTH)

..... FIND LINK TO PUT VEHICLES ON.
DO 1100 KTH=1,NUMLKS
IF(NONXT(NTH+1) .EQ. LKID(KTH+1) .AND. NOIDNO(NTH) .EQ. LKID(KTH+1))
   GO TO 1110
1100 CONTINUE
CALL DUMP(NTH,'SETUP • •• 1100C»*1>
1110 NOOUTP(NTH) = KTH
GO TO 2000

..... INTERNAL NODES
1500 CONTINUE
ITH = NOSIG(NTH)
IF(ITH .EQ. 0) GN TO 2000
IDNTH = NOIDNO(NTH)

..... TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS
NODE.
DO 1600 KTH=1,NUMLKS
IF(LKID(KTH+2) .NE. IDNTH) GO TO 1600

..... OK, NOW FIND OUT WHERE IT CAME FROM.
DO 1520 I=1,4
1520 CONTINUE
CALL DUMP(NTH,'SETUP • •• 1520C»*1>
1525 SIGINF(ITH+1) = KTH
1600 CONTINUE
2000 CONTINUE

..... FOR EACH LINK FIND THE DESTINATIONS.
DO 3000 KTH=1,NUMLKS
DO 2500 I=1,3
  IF(LKDEST(KTH,I) .EQ. 0) GO TO 2500
  DO 2400 KTHCHK=1,NUMLKS
  IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,I)) GO TO 2400
  IF(LKID(KTHCHK,1) .NE. LKID(KTH+2)) GO TO 2400
  LKDEST(KTH+1) = KTHCHK
  GO TO 2500
2400 CONTINUE
2500 CONTINUE
3000 CONTINUE
RETURN
END

SUBROUTINE SIGCHK

C
C THIS SURROUTINE CHECKS THE SIGNAL SETTINGS TO
C SEE IF A CHANGE IS NECESSARY.
C
INCLUDE DEFINS
DO 2000 ITH=1,NUMSIG
DO 1999 I=1,2

C
C..... FIRST CHECK THE SIGNAL'S CLOCK.
C
IF(SIGCLK(ITH,I) .GT. CLOCK) GO TO 1999
C
C..... BRANCH BASED ON OLD STATE.
C
IF(SIGSTA(ITH,I)=-1) 1100,1200,1300

1100 SIGSTA(ITH,I) = 1
SIGCLK(ITH,I) = SIGCLK(ITH,I) + AMBERT
GO TO 1999
C
1200 SIGSTA(ITH,I) = 2
SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGRED(ITH,I)
GO TO 1999
C
1300 SIGSTA(ITH,I) = 0
SIGCLK(ITH,I) = SIGCLK(ITH,I) + SIGGRN(ITH,I)
C
C..... CHECK QUEUE LENGTH FOR THE APPROACHES THAT HAVE CHANGED.
C
DO 1500 IP=1,2
II = I + 2 * (IP - 1)
KTH = SIGINP(ITH,II)
C
C..... SET UP AND CHECK FIRST LANE.
C
    IB1 = LKFSTB(KTH)
    IB2 = IB1 + LKLNK(KTH) - 1
    LQ = 0
    DO 1400 IR=IB1,IB2
      IF(BLKARY(IR) .LT. CAP) GO TO 1410
    LQ = LQ + CAP
1400 CONTINUE
IB = IB2 + 1
IF(LKLNG(KTH) .EQ. 2) IB = IR + LKLNG(KTH)
1410 IF(BLKARY(IB) .LT. CAP / 2.0) GO TO 1420
LQ = LQ + BLKARY(IB)
1420 IF(LQ .GT. LKMAXQ(KTH,1)) LKMAXQ(KTH,1) = LQ
IF(LKLANS(KTH) .EQ. 1) GO TO 1500
C
C..... IF TWO LANES, CHECK SECOND LANE ALSO.
C
IB1 = IB2 + 1
IB2 = IB2 + LKLNG(KTH)
LQ = 0
DO 1450 IR=IB1,IB2
IF(BLKARY(IR) .LT. CAP) GO TO 1460
LQ = LQ + CAP
1450 CONTINUE
IB = IB2 + 2
1460 IF(BLKARY(IB) .LT. CAP / 2.0) GO TO 1470
LQ = LQ + BLKARY(IB)
1470 IF(LQ .GT. LKMAXQ(KTH,2)) LKMAXQ(KTH,2) = LQ
1500 CONTINUE
1999 CONTINUE
2000 CONTINUE
C**** WRITE(6,1) CLOCK,((SIGSTA(I1,I2),I2=1,2),I1=1,NUMSIG)
1 FORMAT(///,'CLOCK **19** SIGNS;',70(1X,2I2))
RETURN
END
SUBROUTINE STAT(K)
C
C THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARMUP AND THEN PRINTS THEM AFTER THE FINISH.
C
INCLUDE DEFINES
C
C..... IF FINISH*GO TO 1000
IF(K .EQ. 2) GO TO 1000
DO 100 KTH = 1,NUMLKS
DO 100 I=1,2
LKUEL(KTH* I) = 0
LKMAXQ(KTH* I) = 0
100 LKVL(KTH* I) = 0
RETURN
C
C..... PRINT THE STATISTICS IN THIS SECTION.
C
1000 TOTIM = FINISH - WARMUP
WRITE(6,1)
1 FORMAT(///,'VEHICLE COUNT,','VEHICLE VOLUME','MAX QS','VEHICLE DELAY','DELAY PER VEHICLE','LINK LANE','LANE 1 LANE 2 TOTAL','LANE 1 LANE 2 TOTAL','LANE 1 LANE 2 TOTAL','/)
DO 1100 KTH = 1,NUMLKS
LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
LKVL1 = LKVL(KTH,1) * 3600 / TOTIM
LKVL2 = LKVL(KTH,2) * 3600 / TOTIM
LKVOLT = LKVL1 + LKVL2
LKDELT = LKDEL(KTH'1) + LKDEL(KTH'2)
DELVL1 = LKDEL(KTH'1) / FLOAT(LKVL(KTH,1))
DELVL2 = LKDEL(KTH'2) / FLOAT(LKVL(KTH,2))
DELVLT = LKDELT / FLOAT(LKVL(KTH,1) + LKVL(KTH,2))

1100 WRITE(6,2) KTH,LKANS(KTH),LKVL(KTH,1),LKVL(KTH,2),LKVTOT
1 LKVL1,LKVL2,LKVL,T,LKMAXO(KTH',I),I=1,2,LKDEL(KTH',1),
2 LKDEL(KTH,2),LKDELT,DELVL1,DELVL2,DELVLT
2 FORMAT(14,4X,12,2(1X,3I9),3X,T,2,13,8,2I9,2X,3F9.2)
5 FORMAT(//,23H INTERSECTION ID NUMBER,14/12H INPUT LINKS,6X,
1 14H VEHICLE VOLUME,10X,9H MAX QUEUE,11X,13H VEHICLE DFLAY,12X,
2 17H DELAY PFR VEHICLE,//,14X,22HLANE 1 LANE 2 TOTAL,4X,
3 14HVLANE 1 LANE 2 TOTAL 2(4X,22HLANE 1 LANE 2 TOTAL,1/)
6 FORMAT(17,4X,18,19,2,8H TOTAL INTERSECTION,11X,13H VOLUME,
1 8H TOTAL INPUT LINKS,6X,40H AVERAGE INTERSECTION DELAY PFR VEHICLE =,
2 2 F7.2)
8 FORMAT(//,32H AVERAGE LINK DELAY FOR SYSTEM = F7.2)
9 FORMAT(29H INTERSECTION OUTPUT SUMMARY,)
WRITE(6,9)
NTOT = 0
NDTOT = 0
DO 1275 NTH=1,NUMNOS
IF(NODETYPE(NTH) LT 3) GO TO 1275
NINT = 0
NDEL = 0
WRITE(6,5) NOIDNO(NTH)
1 T = NOSIG(NTH)
DO 1250 II=1,4
KTH = SIGINP(T,II)
LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
NINT = NINT + LKVTOT
LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
NDEL = NDEL + LKDELT
DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
DELVLT = LKDELT / FLOAT(LKVTOT)
1250 WRITE(6,6) KTH,LKVL(KTH,1),LKVL(KTH,2),LKVTOT,LKMAXO(KTH,1),
1 LKMAXO(KTH,2),LKDELT,DELVL1,DLVL2,DELVLT
1275 CONTINUE
END
SUBROUTINE STATA(I)
C
C PERIOD BY PERIOD OUTPUT GENERATOR.* LKSOUT IS AN ARRAY
C TELLING THE DESIRED OUTPUT LINKS.
C
INCLUDE DEFINES
DIMENSION LKSOUT(4),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
1 DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
DATA LKSOUT / 1,3,0,0 /
PARAMETER N=2
GO TO (10,20),I
10 CONTINUE
C**** WRITE(6,4)
DO 15 II=1,N
NLKII)=0
NSII)=0
NRII)=0
LII)=0
DEL1II)=0
LII)=0
LII)=0
15 LRKII)=0
RETURN
C**** WRITE(6,1) CLOCK,NUMVEH
20 CONTINUE
DO 30 II=1,N
MM = LKSOUT(II)
LIP = LKVU(MM,1) + LKVU(MM,2)
NII)=LIP - LII)
LII)=LIP
IDELP = LKDEL(MM,1) + LKDEL(MM,2)
DELYI)=IDELP-DFL1(II)
IDELII)=IDELP
AVGII)=DELYI)/NII)
XII)=IDELP/FLOAT(LIP)
C**** WRITE(6,2) LIP,
C****1 IDELP,X1(II),N1(II),NLII),NSII),NR1(II),AVG1(II)
30 CONTINUE
PUNCH 3,CLOCK,NUMVEH,((NII),AVGII),XII),II=1,N)
RETURN
1 FORMAT(//, 'VEGEN SUBROUTINE')
FUNCTION TBAGEN(NTH)
C THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
AT THE NTH NODE. AN EXPONENTIAL TIME BETWEEN
ARRIVALS IS ASSUMED.
C
INCLUDE DEFINS
TBAGEN = -ALOG(RNN1(NOISEED(NTH)))*NOPARS(NTH,1)
RETURN
END
SUBROUTINE VEGEN
C THIS SUBROUTINE CHECKS THE CLOCK AT EACH GENERATE
NODE TO SEE IF IT IS TIME TO GENERATE ANOTHER
VEHICLE.
C
INCLUDE DEFINS
C**** WRITE(6,1)
1 FORMAT(//,' VEGEN SUBROUTINE')
DO 2000 NTH=1,NUMNOS
  IF(NOTYPE(NTH) .GT. 1) GO TO 2000

  ***** CHECK TO SEE IF READY FOR ANOTHER VEHICLE
  1000 IF (CLOCK .LT. NOCLK(NTH)) GO TO 2000

  ***** FIND OUT IF HOLDING AREA IS AVAILABLE.
  KTH = NOOUTP(NTH)
  IB = IBHOLD(KTH)
  IF(IB .EQ. 0) GO TO 2000

  ***** MOVE A VEHICLE INTO THE AREA.
  1900 BLKARY(IB) = RLKARY(IR) + 1
  NUMVEH = NUMVEH + 1

  ***** COMPUTE TIME OF NEXT ARRIVAL
  NOCLK(NTH) = NOCLK(NTH) + TBAGFN(NTH)

  **** WRITE(6,2) NTH,KTH,IB,NUMVEH
  2 FORMAT (NTH,'I4', KTH,'I4', IB,'I4', NUMVEH,'I5')

  ***** READY FOR ANOTHER*
  GO TO 1000

2000 CONTINUE
RETURN
END
APPENDIX E

FORTRAN LISTING OF NEXT MODEL
DEFINS PROCEDURE

C SETUP FOR NODES.
C PARAMETER MAXNO=35
REAL NOPARS,NOLJ
INTEGER FSTSQ
COMMON /NODES/ NOTDNO(MAXNO),NOTYPE(MAXNO),NONXT(MAXNO+4),
1 NOSEED(MAXNO),NOSETX(MAXNO),NOPARS[MAXNO,2],NOOUTP[MAXNO],
2 NOGEOM[MAXNO],FSTSQ(MAXNO),NOIG(MAXNO),NOCLK(MAXNO),NUMNOS

C SETUP FOR SIGNALS.
C PARAMETER MAXSIG=20
INTEGER SIGCYC,SIGNR,SIGNRED,SIGOFF,SIGNXT,SIGSTA,SIGCLK,SIGNP
1 AMB
COMMON /SIG/ SIGCYC(MAXSIG),SIGSTA(MAXSIG),SIGOFF(MAXSIG),
1 SIGRED(MAXSIG),SIGCLK(MAXSIG),
2 SIGNT,SIGNP(MAXSIG),SIGCM(MAXSIG),NUMSIG,AMB

C SETUP FOR LINKS.
C PARAMETER MAXLKS=100
REAL LKPROB
COMMON /LINKS/ LKTD(MAXLKS,2),LKLNG(MAXLKS),LKLANS(MAXLKS),
1 LKNOD(MAXLKS),LKLPOS(MAXLKS),LKA(RMAXLKS),LKCAP(MAXLKS),
2 LKCON(MAXLKS),LKPROB(MAXLKS,2),LKDST(MAXLKS,3),NMLKS,
3 LKDRL(MAXLKS),LKDRLD(MAXLKS),LK(MAXLKS,2),LKDSTK(MAXLKS,2),
4 LKSTOP(MAXLKS,2),LKV(MAXLKS,2),LKV(MAXLKS,3),LKMAXQ(MAXLKS,2),
5 LKLEFT(MAXLKS),LKA(RV(MAXLKS,2),LKDTS(MAXLKS,2)

C SETUP FOR VEHICLES.
C PARAMETER MAXVEH=2000
INTEGER VLKTIM,VLK,VLAN,VTURN,VDSP,VTIME,VSTATE,VEHNXT
COMMON /VEH/ VDSP(MAXVEH),VLK(MAXVEH),VLAN(MAXVEH),VTURN(MAXVEH),
1 VTIME(MAXVEH),VSTATE(MAXVEH),VEHNXT(MAXVEH),VLKTIM(MAXVEH),
2 NUMVEH

C SETUP FOR INTERSECTIONS.
C PARAMETER MAXLKS=200
INTEGER BLKARY
COMMON /INT/ BLKARY(MAXBLK),NRLKS

C SETUP FOR CHAINS.
C INTEGER SIGTIM,SIGN,VEGTM,VEGIND,VEGFLG
COMMON /CHAIN/ VEGTIM,VEGIND,VEGFLG,MOVST,MOVIM,SIGTIM,SIGN

C GENERAL COMMON VARIABLES.
C INTEGER CLOCK,FINISH,WARMUP
COMMON CLOCK,SEED,FINISH,WARMUP

* * * * * * * * * * * * * * * * * * *
**DEFINITION OF VARIABLES USED IN THE PROGRAM.**

- **AMBERT** - THE LENGTH OF AMBER TIME.
- **BLKARY(NTH)** - GENERAL ARRAY OF INTERSECTION BLOCKS.
- **CLOCK** - MASTER SIMULATION CLOCK.
- **FINISH** - INITIALLY THE AMOUNT OF TIME TO GATHER STATISTICS.
  - THEN CHANGED TO SIMULATION FINISH TIME.
- **FSTSQ(NTH)** - INDEX TO FIRST SQUARE OF INTERSECTION.
- **ISEED** - THE RANDOM NUMBER SEED.
- **IPTH** - INDEX USED TO POINT TO SPECIFIC SIGNAL.
- **LKARM(KTH)** - APPROACH ARM (1-4) OF LINK TO INTERSECTION.
- **LKCAP(KTH)** - CAPACITY OF KTH LINK IN VEHICLES.
- **LKCONT(KTH)** - CONTENTS OF KTH LINK.
- **LKDELD(KTH,1-3)** - CUMULATIVE DELAY BY DESTINATIONS.
- **LKDEST(KTH,1-3)** - LINK DESTINATIONS: LEFT, STR, RIGHT.
- **LKID(KTH,1-2)** - SYMBOLIC I.D. OF TAIL (1) AND HEAD (2).
- **LKLANS(KTH)** - NUMBER OF LANES.
- **LKLLEFT(KTH)** - FLAG INDICATING AN EXISTING LEFT TURN DELAY.
- **LKLNG(KTH)** - LINK LENGTH (IN FEET).
- **LKMXX(KTH,1-2)** - CURRENT MAX QUEUE BY LANES.
- **LKNOD(KTH)** - INDEX TO NODE AT HEAD OF LINK.
- **LKPOS(KTH)** - LINK APPOSING A LEFT TURN.
- **LKPROB(KTH,1-2)** - CUMULATIVE PROB OF LEFT & STR. MOVE.
- **LKV(KTH,1-2)** - CURRENT QUEUE LENGTH BY LANES.
- **LKQCLK(KTH,1-2)** - TIME OF LAST DEPARTURE.
- **LKSTOP(KTH,1-2)** - CUMULATIVE NUMBER OF STOPS.
- **LKVD(KTH,1-2)** - LINK VEHICLE COUNT BY DESTINATION.
- **LKVL(KTH,1-2)** - LINK VEHICLE COUNT BY LANES.
- **MAXBLK** - MAXIMUM NUMBER OF INTERSECTION BLOCKS.
- **MAXLKS** - MAXIMUM POSSIBLE NUMBER OF LANES.
- **MAXNO** - MAXIMUM POSSIBLE NUMBER OF NODES.
- **MAXSIG** - MAXIMUM POSSIBLE NUMBER OF SIGNALS.
- **MAXVEH** - MAXIMUM POSSIBLE NUMBER OF VEHICLES.
- **MOVFS** - FIRST VEHICLE ON THE MOVE CHAIN.
- **NOCLK(NTH)** - ARRIVAL TIME OF NEXT VEHICLE AT NODE.
- **NOEGOM(NTH)** - NODE GEOMETRY: 1 = 2X2, 3 = 4X4.
- **NOINPUT(NTH)** - INDEX TO LINKS TAKING FROM THIS NODE.
- **NODPARS(NTH)** - PARAMETERS FOR GENERATE NODES.
- **NOSEED(NTH)** - RANDOM NUMBER SEED FOR GENERATION.
- **NOSIG(NTH)** - INDEX TO SIGNAL FOR NTH NODE.
- **NOTYPE(NTH)** - NODE TYPE:
  - 0 - BOTH GENERATE AND TERMINATE.
  - 1 - GENERATE ONLY.
  - 2 - TERMINATE ONLY.
  - 5 - FIXED TIME CONTROLLED.
- **NTH** - INDEX USED TO POINT TO SPECIFIC NODE.
**NUMLKS** - NUMBER OF LINKS IN MODEL.
**NUMNOS** - NUMBER OF NODES.
**NUMSIG** - NUMBER OF SIGNALS.
**NUMVEH** - NUMBER OF VEHICLES IN NETWORK.
**SIGCLK(ITH, 1-2)** - TIME OF NEXT CHANGE OF STATE.
**SIGCYC(ITH)** - CYCLE LENGTH OF ITH SIGNAL.
**SIGGRN(ITH, 1-2)** - GREEN TIME FOR ITH SIGNAL.
**SIGIND** - POINTER TO NEXT SIGNAL TO CHANGE.
**SIGNP(ITH, 1-4)** - LINKS POINTING AT ITH SIGNAL.
**SIGNXT(ITH, 1-2)** - NEXT SIGNAL ON THE EVENTS CHAIN.
**SIGOFF(ITH, 1-2)** - SIGNAL OFFSET TIME.
**SIGRED(ITH, 1-2)** - RED TIME FOR ITH SIGNAL.
**SIGSTA(ITH, 1-2)** - STATE OF ITH SIGNAL:
  0 - GREEN,
  1 - AMBER,
  2 - RED.
**SIGTIM** - TIME OF NEXT SIGNAL CHANGE.
**VDSP(JTH)** - VEHICLE'S DESIRED SPEED.
**VEGFLG** - FLAG INDICATING A DELAYED ENTRY TO PROCESS.
**VEGIND** - INDEX TO FIRST NODE ON THE GENERATE CHAIN.
**VEGTIM** - EVENT TIME OF THE NEXT UNBLOCKED ARRIVAL.
**VEHNXT(JTH)** - NEXT VEHICLE ON MOVEMENT CHAIN.
**VLAN(JTH)** - LANE DESIRED.
**VLK(JTH)** - INDEX OF CURRENT OR MOST RECENT LINK.
**VLKTIM(JTH)** - CLOCK TIME VEHICLE ENTERED LINK.
**VSTATE(JTH)** - THE STATE OF THE JTH VEHICLE:
  0 - ON LINK ARRIVING AT INTERSECTION,
  1 - IN QUEUE WAITING (BUT NOT AT HEAD),
  2 - AT HEAD OF QUEUE WAITING FOR SIGNAL,
  3 - AT HEAD OF QUEUE IN BLOCKED STATE,
  4 - IN S1,
  5 - IN S2,
  6 - IN S3.
**VTIME(JTH)** - NEXT EVENT TIME FOR JTH VEHICLE.
**VTURN(JTH)** - TURN INDICATOR:
  1 - LEFT,
  2 - STRAIGHT,
  3 - RIGHT.
**WARMUP** - WARM UP TIME TO GAIN STATISTICAL EQUIL.

END

**THIS IS THE MAIN PROGRAM FOR THE NEXT EVENT MODEL.**
**ONE CLOCK UNIT EQUALS ONE TENTH OF A SECOND.**
INCLUDE DEFIN.List

C

...... INITIALIZATION
NTH = 0
MOVTIM = 2**30
NBLKS = 0

C

...... RUN PARAMETERS

READ(5,1) IRUN, INET, AMBERT, ISEFD, FINISH, WARMUP
1 FORMAT(
MODEL = 'NEXT'
WRITE(21) MODEL, IRUN, INET, ISEFD
WRITE(22) MODEL, IRUN, INET, ISEED
PUNCH 6: IRUN, INET, ISEED
6 FORMAT('NEXT MODEL IRUN = ', I4, ' INFT = ', I4, ' ISFED = ', I12)
CLOCK = -10
AMBERT = AMBERT * 10
WARMUP = WARMUP * 600
FINISH = FINISH * 600 + WARMUP
WRITE(6,2)
2 FORMAT('************************ NEXT MODEL******************************', //)
3 FORMAT(19///, 'FINISH ', I8, ' WARMUP ', I8)

C

...... CALL INPUT SUBROUTINES.

CALL EXTERN(NTH)
CALL INTERN(NTH)
CALL INPLKS

C

...... CALL SETUP TO FINISH DATA MANIPULATION.

CALL SETUP
CALL DUMP(0, 'DATA CHECK ', n)

C

...... THE SIMULATION.

1000 ICLOCK = MIN(VEGTIM, SIGTIM, MOVTIM)
1100 IF(ICLOCK .LE. CLOCK) CALL DUMP(ICLOCK, 'ICLK = ', CLOCK)
1100 CALL SETUP
1100 IF(MOD(CLOCK, 900) .EQ. 0) CALL STA(2)
1100 WRITE(22) CLOCK, NUMVEH

C**** IT1 = ITIME(IT2, IT3)
C**** XTIME = IT1 * .0002
C**** WRITE(6,4) CLOCK, NUMVEH, XTIME

C

4 FORMAT('CLOCK = ', I4, ' NUMVEH = ', I4, ' CPU TIME REMAINING = ', F5.2)
4 IF(CLOCK .GE. WARMUP) GO TO 2000
4 IF(CLOCK .LT. SIGTIM) GO TO 1100
4 CALL SIGCHK

1100 IF(CLOCK .LT. VEGTIM) GO TO 1200
1100 CALL VEGEN

1200 IF(CLOCK .LT. MOVTIM) GO TO 1300
1200 CALL MOVVEH
1200 IF(VEGFLG .EQ. 1) CALL VEGEN
1300 GO TO 1000
C
C . . . . WARMUP OVER, CLEAR STATISTICS.
C
2000 CALL STAT(1)
   CALL STATA(1)
   GO TO 2200
2100 ICLOCK = MIN(VEGTM, SIGTIM, MOVTIM)
   IF(ICLOCK .LE. CLOCK) CALL DUMP(ICLOCK, 'ICLK = ', CLOCK)
   CLOCK = ICLOCK
   IF(MOD(CLOCK, 900) .EQ. 0) CALL STATA(2)
   WRITE(22) CLOCK, NUMVEH
C**** IT1 = ITIME(IT2, IT3)
C**** XTIME = IT1 * .0002
C**** WRITE(6*4) CLOCK, NUMVEH, XTIME
   IF(CLOCK .GT. FINISH) GO TO 3000
2200 CONTINUE
   IF(CLOCK .LT. SIGTIM) GO TO 2300
   CALL SIGCHK
2300 IF(CLOCK .LT. VEGTM) GO TO 2400
   CALL VEGEN
2400 IF(CLOCK .LT. MOVTIM) GO TO 2500
   CALL MOVVEH
   IF(VEGFLG .EQ. 1) CALL VEGEN
2500 GO TO 2100
C
C . . . . SIMULATION OVER, PRINT STATISTICS.
C
3000 CALL STAT(2)
   CALL CHAIN
   ENDFILE 21
   ENDFILE 22
   STOP
   END
   SUBROUTINE CHAIN

   . . . . THIS SUBROUTINE PRINTS THE VEHICLE CHAIN IN THE
   . . . . PROPER SEQUENCE.
   INCLUDE DEFINS
   JTH = MOVFSF
   WRITE(6*1) NUMVEH
   10 IF(JTH .EQ. 0) GO TO 20
      WRITE(6*2) JTH, VEHNX(JTH), VEHTME(JTH), VDSP(JTH), VSTATF(JTH),
      1 VLK(JTH), VLAN(JTH), VTURJ(JTH), VLKTM(JTH)
      JTH = VEHNX(JTH)
   GO TO 10
20 WRITE(6*3) (LKCONT(KTH), KTH=1, NUMLKS)
   RETURN
   1 FORMAT(' VEHICLE CHAIN = NUMVEH', '15/2', ' JTH ', NVT, 'TIME',
      1 'DSP: ', 'STAT', 'LINK', 'LANF', 'TURN', 'LKTIME')
   2 FORMAT(9(1X, 16))
   3 FORMAT('LINK CONTENTS:', 2014)
   END
   SUBROUTINE DUMP(M*10, N2*K)
C
C . . . . THIS SUBROUTINE PROVIDES A DUMP OF THE VARIABLES.
IN CASE OF AN ERROR, THE VALUE OF K DETERMINES
THE TYPE OF DUMP:
0 = SFT UP CHK ON NODES 
AFTER PRINTOUT ON NODES, FTC,
1 = ERROR ON INPUT,
> 1 = ERROR WHILE RUNNING - LMK NUMBER.

INCLUDE DEFINS

***** GENERAL INFORMATION DUMP.
WRITE(6,1) K
1 FORMAT(/** ERROR DUMP OUTPUT. K = ',I5 )
100 WRITE(6,3) M,N,P
3 FORMAT(' THING ',I5,' NOW BEING PROCESSED. ',/,' MESSAGE ='
1 '1x,2A6')

***** NODE OUTPUT.

1001 MAX = MAXNO
WRITE(6,1010) MAX,NUMNOS
1010 FORMAT(///, ' NODE OUTPUT MAX = 'I2,12X,
1 ' NUMBER = 'I2,12X, ' INDEX ID NEXT NODE ID',
2 ' TYPE GEOM INDEX LINKS IN LINK OUT SIGNAL',
3 ' FSTB CLOCK PARAMETERS GEN NXT',//)
DO 101 I=1,NUMNOS
101 WRITE(6,1011) I,NODNO(I),(NONXT(I,K),K=1,4),NOTYPE(I),NOGEOM(I),
1 NOOUTP(I),NOSIG(I),FST5Q(I),NOCKL(I),
2 (NOPARS(I,K),K=1,2),NOGNXT(I)
1011 FORMAT(I4,6X,I2,3X,4I3,6X,I2,5X,I2,2X,16X,I2,8X,I2,5X,
1 13,F8.0,2X,F5.0,1X,F5.0,5X,13,F8.0)

***** SIGNAL OUTPUT

MAX = MAXSIG
LAM = AMBERT
WRITE(6,1120) MAX,NUMSIG,LAM
1120 FORMAT(///, ' SIGNAL OUTPUT MAX = 'I2,' NUMBER = '
1 '12,' AMBER TIME = 'I2,12X,23X, ' MAJOR',
2 ' ** ** ** **22X,** MINOR ** ** */,
3 ' INDEX CYCLE,2(6X,1 OFFSET GREEN RED STATE CLOCK NXT ',
4 '))
DO 102 I=1,NUMSIG
102 WRITE(6,1121) I,STGCYC(I),(SIGOFF(I,K),SIGGRN(I,K),SIGRED(I,K),
1 SIGSTA(I,K),SIGCLK(I,K),SIGNXT(I,K),K=1,2)
1021 FORMAT(I4,4X,I4,2(8X,13,5X,13,4X,13,5X,12,3X,15,14,2X))

***** LINK OUTPUT.

MAX = MAXLKS
WRITE(6,1130) MAX,NULMS
1130 FORMAT(///, ' LINK OUTPUT MAX = 'I3,' NUMBER = 'I3',//,
1 ' INDEX ID NO. LENGTH LANES NO. LEFT LKCAP LKCONT 01',
2 ' DESTINATIONS PROBABILITIES ARM LEFT LKCAP LKCONT 01 ',
3 ' G2',//)
DO 103 I=1,NULMS
103 WRITE(6,1131) I,LKID(I,1),LKDNO(I,1),LKNOD(I),LKNOD(I),
1 LKOPOS(I),(LKDST(I,K),K=1,3),(LKPNO(I,K),K=1,2)
2 LKARM(I) + LKLEFT(I) + LKCAP(I) + LKCONT(I) + LKO(I,1) + LKO(I,2)
1031 FORMAT(5, + 4X + P13, + 17, + 6X + I1, + 6X + P7, + 7X + I3, + 3X + 2I6, + 3X + I3, + 2X, + 2F7, + 3X + I1, + 5X + I1, + 5X + I3, + 6X + I2, + 2X + 2I4)
IF(K .EQ. 0) RETURN
IF(K .EQ. 1) STOP

C C

..... BLOCK OUTPUT

MAX = MAXBLK
WRITE(6, 1040) MAX, NBLKS
1040 FORMAT(1, + BLOCK OUTPUT MAX = *, I4, + NUMBER = *, I4, + //,
1 + 'FIRST VALUE .....................................' + //
I1 = -24
104 I1 = I1 + 25
I2 = I1 + 24
WRITE(6, 1041) I1, (BLKARY(I)), IT=I1, I2)
1041 FORMAT(1, + , **, 25T4)
IF(I2 .LT. NBLKS) GO TO 104
C C

..... VEHICLE OUTPUT.

MAX = MAXVEH
WRITE(6, 1050) MAX, NUMVEH
1050 FORMAT(1, + VEHICLE OUTPUT MAX = *, I4, + NUMBER = *, I4, + //,
1 + 'INDEX DSP ASP LINK LANE TURN LKTIME NEXT TIME',
2 + 'STATE', //)
DO 105 I = 1, MAXVEH
IF(VDSP(I) .EQ. 0) GO TO 105
WRITE(6, 1051) I, VDSP(I), VLK(I), VLAN(I), VTURN(I), VLT(M(I))
1 VEHNXT(I), VTIME(I), VSTATE(I)
1 CONTINUE
CALL CHAIN
C C

..... CHAIN OUTPUT.

WRITE(6, 1060) VEGND, VEGTIM, VEGFLG, MOVFST, MOVTIM, SIGIND, SIGTIM
1060 FORMAT(1, + VEGND = *, I5, + VEGTIM = *, I7, + VEGFLG = *, I1, //,
2 + MOVFST = *, I5, + MOVTIM = *, I7, //,
2 + SIGIND = *, I5, + SIGTIM = *, I7)
STOP
END
SUBROUTINE EXTERN(NTH)
C C

..... THIS SUBROUTINE READS THE DATA CARDS ON EXTERNAL
NODES ONLY. CONTROL IS RETURNED TO THE CALLING
PROGRAM AFTER A BLANK CARD IS FOUND

INCLUDE DEFINS

INTEGRAL ALPHA
1000 NTH=NTH+1
READ(5,101) ALPHA, NOIDNO(NTH), NTYPE(NTH), NONXT(NTH),
1 NOPARS(NTH+1), NOPARS(NTH+2), NOGEOF(NTH)
IF(ALPHA , EQ. 1H ) GO TO 2000
GO TO 1000
2000 NTH=NTH+1
RETURN
FUNCTION IDELAY(JTH, KTH, NTH)
      C THIS FUNCTION CALCULATES THE AMOUNT OF DELAY.
      INCLUDE DEFS
      DIMENSION INTERS(3)
      DATA INTERS /3, 2, 1, 0, 0, 5, 2, 1/
      N = NOGEOM(NTH)
      K = VTURN(JTH)
      IDELAY = (CLOCK - VLTIM(JTH) - LKLNG(KTH) * 10 / VDSP(JTH) - INTERS(K * N) * 10) / 10
      WRITE(21) CLOCK, KTH, JTH, IDELAY
      RETURN
      END

FUNCTION IGAP(KTH)
      C THIS FUNCTION TESTS THE GAP THE VEHICLE IS OBSERVING.
      A 0 MEANS GO AND A 1 MEANS STOP.
      INCLUDE DEFS
      DIMENSION PROB(20), AGAP(2)
      DATA PROB / -1.0, -1.0, -0.02, 0.02, 0.10, 0.22, 0.36, 0.50, 0.61, 0.74, 1.0, 0.81, 0.94, 0.90, 0.92, 0.95, 0.96, 0.97, 0.98, 0.99, 0.9995, 0.9999/
      C ARE THERE CARS ON THIS LINK TO CHECK.
      IF (LKCONT(KTH) .LT. 0) GO TO 2000
      C CHECK THE SIGNAL FIRST.
      NTH = LKNOD(KTH)
      ITH = NOSIG(NTH)
      I = 2 - MOD(LKARM(KTH), 2)
      IF (SIGSTA(ITH, I) .NE. 0) GO TO 2000
      C CHECK GAP IN EACH LANE OF LINK.
      AGAP(1) = 1000.0
      AGAP(2) = 1000.0
      DO 300 I = 1, 2
      IF (LKQ(KTH, I) .EQ. 0) GO TO 100
      C QUEUE EXISTS, CHECK TO SEE IF LFAD CAR IS DELAYED.
      JTH = MOVFST
      10 IF (VLK(JTH) .NE. KTH) GO TO 50
      IF (VSTATE(JTH) .GT. 3) GO TO 50
      IF (VSTATE(JTH) .LT. 2) GO TO 50
      IF (VLAN(JTH) .EQ. 1) GO TO 60
      50 JTH = VEHNXT(JTH)
      IF (JTH .EQ. 0) CALL DUMP(KTH, IGAP L, 'KPOS', 33)
      GO TO 10
      60 IF (VTIME(JTH) .GE. CLOCK) GO TO 1000
      GO TO 200
      C NO QUEUE SO LOOK FOR THF FIRST VEHICLE.
C
100 JTH = MOVFST
110 IF(VLK(JTH) .NE. KTH) GO TO 150
   IF(VSTATE(JTH) .NE. 0) GO TO 150
   IF(VLAN(JTH) .EQ. 1) GO TO 160
150 JTH = VEHNXT(JTH)
   IF(JTH .EQ. 0) GO TO 200
   GO TO 110
160 AGAP(I) = VTIME(JTH) - CLOCK
200 IF(LEKLANS(KTH) .EQ. 1) GO TO 301
300 CONTINUE
301 GAP = MIN(AGAP(1), AGAP(2)) / 10.0

C
   ***** LOOK AT SMALLEST GAP AND COMPARE TO TABLE.*
   IF(GAP .LE. 20.0) GO TO 2000
   IF(GAP .LE. 2) GO TO 1000
   INDEX = GAP * 0.5
   IF(RN1(ISEXED) .LE. PRB(INDEX)) GO TO 2000

C
   ***** REJECT.*
1000 IGAP = 1
   RETURN

C
   ***** ACCEPT.*
2000 IGAP = 0
   RETURN
   END
SUBROUTINE INPLKS

C
   ***** THIS SUBROUTINE READS CARDS FOR ALL NETWORK LINKS.
   AFTER ALL DATA CARDS HAVE BEEN READ THE ROUTINE
   CALCULATES SOME CROSS REFERENCE INDICES.
   INCLUDE DEFINES
   ***** THIS IS THE LOCAL DECLARATION.
INTEGER ALPHA
KTH=0
1000 KTH=KTH+1
   READ(5,101) ALPHA,LKIN(KTH),LKLNG(KTH),LKLANS(KTH),
   LKID(KTH,1),LKD(KTH,2),LKID(KTH,3),LKD(KTH,4),LKLPOS(KTH),
   LKPROB(KTH,1),LKPROB(KTH,2),LKD(KTH,1),LKD(KTH,2),LKD(KTH,3)

C
   ***** IF IT IS A RANK CARDS, GO TO 2000.
   IF(ALPHA .NE. 1H ) GO TO 2000

C
   ***** CALCULATE THE CAPACITY OF THE LINK.
   LKCAP(KTH) = LKLANS(KTH) * LKLNG(KTH) / 20.0

C
   ***** FIND NONE NUMBER OF THE HEAD NONE.
DO 1010 NTH=1,NUMNOS
   IF(NOIDNO(NTH) .NE. LKID(KTH,2)) GO TO 1015
1010 CONTINUE
   CALL DUMP(KTH,'INPLKS','DO1010','I')
201

1015 LKNOD(KTH) = NTH
   DO 1016 I=1,4
      IF(LKID(KTH,I) .EQ. NONXT(NTH,I)) GO TO 1017
1016 CONTINUE
   CALL DUMP(KTH,'INPLKS','D01016','1')
1017 LKARM(KTH) = 1

C
..... SET UP THE TURNING PROBABILITIES.
C
1060 LKPROB(KTH*2) = LKPROB(KTH*1) + LKPROB(KTH*2)
   IF(LKPROB(KTH*2) .GE. 0.9985) LKPROB(KTH*2) = 1.0
   GO TO 1000
C
..... FIND LINK OPPOSED TO LEFT TURN.
C
2000 NUMLKS=KTH-1
   IF(NUMLKS .GE. MAVLKS) CALL NIMP(NUMLKS,'INPLKS','MAVLKS','1')
   DO 2500 KTH=1,NUMLKS
      IF(LKOPOS(KTH) .NE. 0) GO TO 2500
      DO 2100 KTH2=1,NUMLKS
         IF(LKID(KTH2,K) .NE. LKID(KTH,K)) GO TO 2100
         IF(LKOPOS(KTH) .NE. LKID(KTH2,K)) GO TO 2100
      2400 CONTINUE
      CALL DUMP(KTH,'INPLKS','LKOPOS','1')
      2450 LKOPOS(KTH) = KTH2
   2500 CONTINUE
RETURN

101 FORMAT(A1,2I3.14.11,I3,2F3.3.313)
END

SUBROUTINE INTERN(NTH)

C
C THIS SUBROUTINE READS DATA CARDS ON INTERIOR NODES ONLY. CONTROL IS RETURNED TO THE CALLING PROGRAM AFTER A BLANK CARD IS FOUND.
C
INTEGER ALPHA,TMP
INCLUDE DEFINES
ITH = 0
1000 NTH = NTH + 1
   READ(5,101) ALPHA,NOIDNO(NTH),NOTYPE(NTH),(NONXT(NTH,I),I=1,4),1
   NGEOM(NTH)
   IF(ALPHA .EQ. 1H ) GO TO 3000
   IF(NOTYPE(NTH) .LT. 4) GO TO 1000
   FSTSQ(NTH) = NBLKS + 1
   NBLKS = NBLKS + 8
C
..... IF NONSIGNA1IZED RETURN TO READING NEXT NODE.
C
   IF(NOTYPE(NTH) .LT. 4) GO TO 1000
C
..... SIGNALIZED INTERSECTIONS REQUIRE ADDITIONAL CARD.
C
   ITH = ITH + 1
   NOSIG(NTH) = ITH
   READ(5,102) ALPHA,KDUM,SIGCYC(ITH),SIGOFF(ITH,1),SIGOFF(ITH,2),SIGGDN(ITH,1),1
   SIGOFF(ITH,2),SIGGRN(ITH,2)
C
..... CALCULATE STATE CHANGE TIMES AND PUT ON CHAIN.
C
SIGCYC(ITH) = SIGCYC(ITH) * 10
DO 2100 I=1,2
SIGOFF(ITH+I) = SIGOFF(ITH+I) * 10
SIGGRN(ITH+I) = SIGGRN(ITH+I) * 10
SIGRED(ITH+I) = SIGCYC(ITH) - SIGGRN(ITH+I) - AMBERT
TEMP = SIGOFF(ITH+I) + SIGGRN(ITH+I)
IF (TEMP GT SIGCYC(ITH)) GO TO 2010
TEMP = TEMP + AMBERT
IF (TEMP GT SIGCYC(ITH)) GO TO 2005
SIGSTA(ITH+I) = 2
SIGCLK(ITH+I) = SIGOFF(ITH+I)
GO TO 2020
2005 SIGSTA(ITH+I) = 1
SIGCLK(ITH+I) = TEMP - SIGCYC(ITH)
GO TO 2020
2010 SIGSTA(ITH+I) = 0
SIGCLK(ITH+I) = TEMP - SIGCYC(ITH)
C
C
C PUT SIGNAL ON EVENTS CHAIN. FIRST CHECK EMPTY CHAIN.
C
C 2020 IF (ITH*I .EQ. 1) GO TO 2090
C
C ***** NEXT SEE IF IT SHOULD BE FIRST.
C
ITHOLD = ABS(SIGIND)
IOLD = 1
IF (SIGIND LE 0) IOLD = 2
IF (SIGCLK(ITH*I) .LE. SIGCLK(ITHOLD*IOLD)) GO TO 20 An
C
C ***** MUST BRACKET BETWEEN AN OLD AND A NEW.
C
2025 ITHNEW = SIGNXT(ITHOLD*IOLD)
IF (ITHNEW) 2050,2030,2035
2030 INEW = 1
GO TO 2075
2035 INEW = 1
GO TO 2055
2050 ITHNEW = -ITHNEW
INEW = 2
C
C ***** IF NEW RIGGED HAVE FOUND SPOT TO INSERT.
C
2055 IF (SIGCLK(ITHNEW,INEW) .GE. SIGCLK(ITH*I)) GO TO 2075
C
C ***** OTHERWISE CHANGE NEW TO OLD AND CHECK NEXT SLOT.
C
ITHOLD = ITHNEW
IOLD = INEW
GO TO 2025
C
C ***** POSITION FOUND PUT ON CHAIN.
C
2075 IF (I .EQ. 1) GO TO 2076
SIGNXT(ITHOLD,IOLD) = -ITH
GO TO 2077
2076 SIGNXT(ITHOLD,IOLD) = ITH
IF(INEW .EQ. 1) GO TO 2078
SIGNXT(I TH*I) = -ITHNEW
GO TO 2100
2078 SIGNXT(I TH*I) = ITHNEW
GO TO 2100

C
C ***** PUT AT HEAD OF CHAIN.
C
2080 SIGNXT(I TH*I) = SIGIND
SIGIND = ITH
IF(I .EQ. 2) SIGIND = -ITH
SIGTIM = SIGCL(K (I TH*I)
GO TO 2100
C
C ***** FIRST SIGNAL ON CHAIN.
C
2090 SIGIND = 1
SIGTIM = SIGCLK(1,1)
C
C ***** END OF DO LOOP.
2100 CONTINUE
C
C ***** GO BACK AND READ DATA ON NEXT NODE.
C
GO TO 1000
C
C ***** SET PARAMETERS.
C
3000 NUMNOS = NTH - 1
IF(NUMNOS .GE. MAXNO) CALL DUMP(NUMNOS, 'INTERN', 'MAXNO', 1)
IF(NBLKS .GT. MAXRLK) CALL DUMP(NBLKS, 'INTERN', 'MAXRLK', 1)
NUMSIG = ITH
IF(NUMSIG .GE. MAXSIG) CALL DUMP(NUMSIG, 'INTERN', 'MAXSIG', 1)
RETURN
C
C ***** FORMAT STATEMENTS
C
101 FORMAT(A1, I3, I2, 4I3, I1)
102 FORMAT(A1, I3, 5I3)
END
SUBROUTINE ISQ(I, KTH, NTH, JTH, IARM, I1, I2)
C
C ***** THIS SUBROUTINE CHECKS THE SQUARE INDICATED BY I.
C IF A SQUARE IS AVAILABLE, THE OUTPUT IS THE BLOCK INDEX. IF UNAVAILABLE, THE OUTPUT IS THE NEGATIVE
C OF THE VEHICLE INDEX.
C
INCLUDE DEFINS
DIMENSION IADD(4,4)
DATA IADD /3,4,2,1,4,2,1,3,2,1,3,4,1,3,4
IF(NGEOM(NTH) .NE. 1) GO TO 200
C
C ***** TWO-BY-TWO.
C
100 I2 = 0
IB = FSTSO(NTH) + (IADD(IARM*I) - 1) * 2
IF(BLKARY(IB) .NE. 0) GO TO 110
II = IB
RETURN
110 II = -BLKARY(IB)
RETURN

****** FOUR BY FOUR.*

200 CONTINUE
    IB = FSTSO(NTH) + (IAND(IARM,1) - 1) * 2
    IF(BLKARY(IB) .NE. 0) GO TO 210
    II = IB
    GO TO 230
210 II = -BLKARY(IB)
230 IB = IB + 1
    IF(BLKARY(IB) .NE. 0) GO TO 240
    I2 = IB
    RETURN
240 I2 = -BLKARY(IB)
RETURN
END

SUBROUTINE ISQOUT(I,*KTH,NTH,JTH,IARM)

****** THIS SUBROUTINE FINDS THE BLOCK THE VEHICLE WAS IN
AND REMOVES THE VEHICLE.

INCLUDE DEFINES
DIMENSION IADD(4,4)
DATA IADD/3,4,2,1,4,2,1,3,2,1,3,4,1,3,4,2/)

IB = FSTSO(NTH) + (IAND(IARM,1) - 1) * 2
IF(BLKARY(IB) .NE. JTH) GO To 15
5 BLKARY(IB) = 0
    IF(I .NE. 3) RETURN
10 IB = IB + 1
    BLKARY(IB) = 0
RETURN
15 IB = IB + 1
    IF(BLKARY(IB) .NE. JTH) CALL DUMP(JTH,'ISQOUT','NOFIND',18)
    BLKARY(IB) = 0
RETURN
END

SUBROUTINE MOVEVEH

****** THIS SUBROUTINE MOVES VEHICLES ON THE VEHICLE MOVE
CHAIN. THE ORDER IN WHICH VEHICLES ARE PROCESSED
DEPENDS ON THEIR STATE AND IS SPECIFIED BY THE
ARRAY CALLED ORDER.*

INCLUDE DEFINES
PARAMETER FSTGAP = 3B
INTEGER DELFAC,DISGAP,TIME,ORDFR(7),DFLAY
DIMENSION ITIME(3,3,3),DISGAP(5)
DEFINE TIME(I,J,K) = ITIME(I,J,K)
DATA DISGAP / 31, 27, 24, 22, 21 /
DATA ITIME /1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
C**** WRITE(6,1) 
 1 FORMAT(' MOVVEH SUBROUTINE.')
C**** CALL CHAIN
  DO 1000 M=1,6
    INDEX = ORDER(M)
C
    ***** SET UP TO START AT HEAD OF CHAIN.
    KGOTO = INDEX + 1
25   JHLS = 0
    JTH = MOVFST
    IF(JTH .EQ. 0) GO TO 1010
C
    ***** CHECK EVENT TIME OF THIS VEHICLE.
50 IF(VTIME(JTH) .GT. CLOCK) GO TO 1000
    JTHNXT = VEHNXT(JTH)
C
    ***** IS THIS VEHICLE IN THE RIGHT STATE.
    IF(VSTATE(JTH) .NF. INDEX) GO TO 975
    KTH = VLK(JTH)
    NTH = LKNOD(KTH)
    ILANE = VLAN(JTH)
C
    ***** IF THIS IS A TERMINATE NODE, SKIP OUT OF THIS AREA.
    IF(NTYPE(NTH) .LT. 3) GO TO 100
C
    ***** BRANCH BASED ON THE STATE OF THE VEHICLE.
    GO TO (100,975,300,400,500,600,700)*KGOTO
100  CONTINUE
C
    ***** AN ARRIVAL, FIRST CHECK THE QUEUE.
    DELFAC = 0
    IF(LKQ(KTH,ILANE) .EQ. 0) GO TO 112
C
    ***** PUT IN QUEUE. NO CHANGE IN CHAIN POSITION.
    LKG(KTH,ILANE) = LKG(KTH,ILANE) + 1
    IF(LKG(KTH,ILANE) .GT. LKMAXQ(KTH,ILANE)) LKMAXQ(KTH,ILANE) =
1     LKG(KTH,ILANE)
    LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1
    VSTATE(JTH) = 1
C**** WRITE(6,3) JTH,KTH,ILANE,VSTATE(JTH)
    GO TO 975
C
C
    ***** BRANCH BASED ON LIGHT STATE.
112  ITH = NOSIG(NTH)
    I2 = 2 - MOD(LKARM(KTH),2)
    IF(SIGSTA(ITH,I2) .LT. 1) 160,120,115
C
    ***** RED LIGHT, RESCHEDULE MOVE EVENT.
206

115 VTIME(JTH) = SIGCLK(I2) + FSTGAP
VTIME(JTH) = VTIMF(JTH) / 10 * 10
GO TO 122
C
C AMBER LIGHT, CHECK FOR A LEFT TURN.
C
120 IF(VTURN(JTH) .EQ. 1) GO TO 125
C
C STRAIGHT AND RIGHT TURNS STOP FOR LIGHT.
C
121 VTIME(JTH) = FSTGAP * SIGCLK(I2) + SIGRED(I2)
VTIME(JTH) = VTIMF(JTH) / 10 * 10
122 IF(DELFA > GT. 0) GO TO 123
LKU(KTH,ILANE) = 1
IF(LKMAXG(KTH,ILANE .EQ. 0) LKMAXG(KTH,ILANE) = 1
LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1
123 VSTATE(JTH) = 2
LKDIS(KTH,ILANE) = 0
C**** WRITE(6,6) JTH,KTH,ILANE,VTIMF(JTH)
GO TO 950
C
C LEFT TURNS WHO WERE BLOCKED WILL TRY TO SNEAK THROUGH.
C
125 IF(INDEX .NE. 3) GO TO 121
KTHNEXT = LKDEST(KTH+1)
IF(LKCAP(KTHNEXT) .LE. LKCONT(KTHNEXT)) GO TO 121
C
C BRANCH ON NUMBER OF LANES.
C
127 CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB,IR2)
IF(IB .LT. 0) GO TO 121
CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
GO TO 130
C
C TWO LANE STREET REQUIRES SPACE IN S1 AND NO LEFT TURN IN S2.
C
127 CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
IF(IB1 .LT. 0 .AND. IR2 .LT. 0) GO TO 121
IB = MIN(IB1,IB2)
IF(IB .GT. 0) GO TO 128
JTHCK = -IB
IF(VTURN(JTHCK) .EQ. 1) GO TO 121
IB = MAX(IB1,IB2)
C
128 CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
IF(IB1 .GT. 0) GO TO 130
IF(VTURN(-IB1) .EQ. 1) GO TO 121
130 IF(IB2 .GT. 0) GO TO 135
IF(VTURN(-IB2) .EQ. 1) GO TO 121
C
C MOVE THE VEHICLE INTO THE INTERSECTION.
C
135  \text{VTIME(JTH)} = \text{CLOCK} + \text{TIME}(1,1,\text{NOGEOM(NTH)}) * 10 + \text{DELFAC} \\
\text{VTIME(JTH)} = \text{VTIME(JTH)} / 10 * 10 \\
\text{GO TO 940}

\text{C} \quad \text{...... GREEN SIGNAL, FIRST CHECK S1 FOR SPACE.}

160  \text{CALL ISQ(1,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)} \\
\text{IF(IB1 .GT. 0 .OR. IB2 .GT. 0) GO TO 165}

\text{C} \quad \text{...... VEHICLE IS BLOCKED, PUT IN STATE 3 AND HOLD THE PRESENT CHAIN POSITION.}

162  \text{VSTATE(JTH)} = 3

\text{C} \quad \text{**** WRITE(6,7), JTH,KTH,ILANE} \\
\text{IF(DELFAC .GT. 0) GO TO 975} \\
\text{LKQ(KTH,ILANE)} = 1 \\
\text{IF(LKMAXQ(KTH,ILANE) .EQ. 0) LKMAXQ(KTH,ILANE) = 1} \\
\text{LKSTOP(KTH,ILANE) = LKSTOP(KTH,ILANE) + 1} \\
163  \text{GO TO 975}

\text{C} \quad \text{...... NEXT CHECK CAPACITY OF EXIT LINK AND PUT IN THE BLOCKED STATE IF LINK IS FULL.}

165  \text{K = VTURN(JTH)} \\
\text{KTNXT = LKDEST(KTH,K)} \\
\text{IF(LKCAP(KTNXT) .LE. LKCONT(KTNXT)) GO TO 162} \\
\text{C} \quad \text{...... OTHERWISE BRANCH ON TURNING MOVEMENT.}

\text{C} \quad \text{GO TO (190,175,176,K)}

\text{C} \quad \text{...... RIGHT TURN TRAFFIC.}

170  \text{IB = MAX(IB1,IB2)} \\
\text{VTIME(JTH)} = \text{TIME}(1,2,\text{NOGEOM(NTH)}) * 10 + \text{DELFAC} + \text{CLOCK} \\
\text{VTIME(JTH)} = \text{VTIME(JTH)} / 10 * 10 \\
\text{GO TO 940}

\text{C} \quad \text{...... STRAIGHT TRAFFIC.}

175  \text{IB = MAX(IB1,IB2)} \\
\text{IF(LKLANS(KTH) .EQ. 2) GO TO 180} \\
\text{CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)} \\
\text{IF(IB1 .GE. 0) GO TO 180} \\
\text{IF(VTURN(-IB1) .EQ. 1) GO TO 162} \\
180  \text{VTIME(JTH)} = \text{TIME}(1,2,\text{NOGEOM(NTH)}) * 10 + \text{DELFAC} + \text{CLOCK} \\
\text{VTIME(JTH)} = \text{VTIME(JTH)} / 10 * 10 \\
\text{GO TO 940}

\text{C} \quad \text{...... LEFT TURN VEHICLES. CHECK FOR LEFT TURN IN FRONT.}

190  \text{IB = MIN(IB1,IB2)} \\
\text{IF(IB .GE. 0) GO TO 192} \\
\text{IF(VTURN(-IB) .EQ. 1) GO TO 162} \\
192  \text{IB = MAX(IB1,IB2)} \\
\text{CALL ISQ(2,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)} \\
\text{IF(IB1 .GE. 0) GO TO 195}
Page missing from thesis
CALL ISOQOUT(2,KTH,NTH,JTH,LKARM(KTH))
DELFAC = 0
IF(VTIME(JTH) .NE. CLOCK) DELFAC = IN
GO TO 900

C

***** LEFT TURN MUST CHECK GAP AND S3
610 IF(IGAP(LKPOS(KTH)) .EQ. 1) GO TO 975
CALL ISOQ(3,KTH,NTH,JTH,LKARM(KTH),IB1,IB2)
 IF(IB1 .LT. 0 .OR. IR2 .LT. 0) GO To 975
BLKARY(IB1) = JTH
 IF(IB2 .GT. 0) BLKARY(IB2) = JTH
C**** WRITE(6,10) JTH,NTH
CALL ISOQOUT(2,KTH,NTH,JTH,LKARM(KTH))
VSTATE(JTH) = 6
DELFAC = 0
IF(VTIME(JTH) .NE. CLOCK) DELFAC = IN
VTIME(JTH) = TIME + 3 + NOGEOM(NTH) * 10 + DELFAC + CLOCK
VTIME(JTH) = VTIME(JTH) / 10 * 10
GO TO 950

C

***** VEHICLE TO EXIT S3. CHECK CAPACITY.
700 KTHNXT = LKDEST(KTH+1)
 IF(LKCONT(KTHNXT) .LE. LKCONT(KTHNXT)) GO TO 975
DELFAC = 0
CALL ISOQOUT(3,KTH,NTH,JTH,LKARM(KTH))
GO TO 900

C

***** TERMINATION SECTION.
800 NUMVEH = NUMVEH - 1
ITURN = VTURN(JTH)
LKVL(KTH,ILANE) = LKVL(KTH,ILANE) + 1
LKVD(KTH,ITURN) = LKVD(KTH,ITURN) + 1
VDSP(JTH) = 0
VEHNXT(JTH) = 0
LKCONT(KTH) = LKCONT(KTH) - 1
C**** WRITE(6,2) JTH,NTH
 IF(JTHLST .EQ. 0) GO TO 810
VEHNXT(JTHLST) = JTHNXT
GO TO 980
810 MOVFST = JTHNXT
GO TO 980

C

***** ADVANCE VEHICLE TO NEW LINK.
900 DELAY = IDELAY(JTH,KTH,NTH) + DELFAC
ITURN = VTURN(JTH)
LKVL(KTH,ILANE) = LKVL(KTH,ILANE) + 1
LKDEL(KTH,ILANE) = LKDEL(KTH,ILANE) + DELAY
LKVD(KTH,ITURN) = LKVD(KTH,ITURN) + 1
LKDELD(KTH,ITURN) = LKDELD(KTH,ITURN) + DELAY
LKCONT(KTH,NXT) = LKCONT(KTH,NXT) + 1
VSTATE(JTH) = 0
C**** WRITE(6,8) JTH,KTHNXT
VLK(JTH,NXT) = KTHNXT
X = RN1(1/SEED)
DO 905 I=1,2
IF(X .LE. LKPROB(KTHNXT,I)) GO TO 91n
905 CONTINUE
VTURN(JTH) = 3
VLAN(JTH) = LKLANS(KTHNXT)
GO TO 920
910 VTURN(JTH) = 1
IF(I .EQ. 2) GO TO 915
VLAN(JTH) = 1
GO TO 920
915 VLAN(JTH) = RN1(SEED) * LKLANS(KTHNXT) + 1
920 VTKTIM(JTH) = CLOCK + DELFAC
VTIME(JTH) = LKLNG(KTHNXT) * 10 / VDSP(JTH) + DELFAC + CLOCK
ILANE = VLAN(JTH)
IF(VTIME(JTH) .LT. LKARVT(KTHNXT,ILANE) + 10) VTIME(JTH) =
LKLARVT(KTHNXT,ILANE) + 10
VTIME(JTH) = VTIME(JTH) / 10 + 10
LKARVT(KTHNXT,ILANE) = VTIME(JTH)
IF(JTHLST .NE. 0) GO TO 925
MOVFST = JTHNXT
GO TO 930
925 VEHNXT(JTHLST) = JTHNXT
930 CALL VCHAIN(JTH,VTIME(JTH))
IF(JTHNXT .EQ. VEHNXT(JTH)) JTHLST = JTH
GO TO 980
C
C ***** MOVE INTO S1*
C
940 BLKARY(IB) = JTH
VSTATE(JTH) = 4
LKQCLK(KTH,ILANE) = CLOCK
WRITE(6,5) JTH,KTH,ILANE,VTIME(JTH)
C
C ***** FIND NEW POSITION ON CHAIN*
C
950 IF(JTHLST .NE. 0) GO TO 955
MOVFST = JTHNXT
GO TO 960
955 VEHNXT(JTHLST) = JTHNXT
960 CALL VCHAIN(JTH,VTIME(JTH))
IF(JTHNXT .EQ. VEHNXT(JTH)) JTHLST = JTH
IF(VSTATE(JTH) .NE. 4) GO TO 980
C
C ***** VEHICLES MOVED TO S1 REQUIRE ADDITIONAL PROCESSING*
C
LKCONT(KTH) = LKCONT(KTH) - 1
IF(DELFAC .NE. 0) LKO(KTH,ILANE) = LKO(KTH,ILANE) + 1
IF(LKO(KTH,ILANE) .GT. 0) GO TO 961
LKDIS(KTH,ILANE) = 0
GO TO 980
961 N = LKDIS(KTH,ILANE) + 1
LKDIS(KTH,ILANE) = N
IF(N .GT. 5) N = 5
JTHLSP = 0
JTHP = MOVFST
962 JTHNXP = VEHNXT(JTHP)
IF(VTIME(JTHP) .GT. CLOCK) CALL DUMP(KTH,MOVHEH,3ERCH,342)
IF(VLK(JTHP) .NE. KTH) GO TO 963
IF(VSTATE(JTHP) .NE. 1) GO TO 963
IF(VLAN(JTHP) .EQ. ILANE) GO TO 965

963 JTHLSP = JTHP
JTHP = JTHNXP
IF(JTHP .EQ. 0) CALL DUMP(KTH,'MOVVEH','QSERCH',348)
GO TO 962

C

***** NEW LEADER IN QUEUE; PULL OUT AND SCHEDULE TO DISCHARGE.
C

965 IF(JTHLSP .EQ. 0) GO TO 967
VEHNXT(JTHLSP) = JTHNXP
GO TO 968

967 MOVFST = JTHNXP
968 VTIME(JTHP) = (CLOCK + DISGAP(N)) / 10 * 10
VSTATE(JTHP) = 3
CALL VCCHAIN(JTHP,VTIME(JTHP))

C**** WRITE(6,4) JTHP,KTH,ILANE,VTIME(JTHP)
GO TO 25

C

C

975 JTHLST = JTH

C

***** CHECK NEXT VEHICLE.
C

980 IF(JTHNXT .EQ. 0) GO TO 1000
JTH = JTHNXT
GO TO 50

C

***** END OF DO LOOP.
C

1000 CONTINUE

C

***** FIND TIME OF NEXT EVENT
C

JTH = MOVFST
1001 IF(JTH .EQ. 0) GO TO 1010
IF(VTIME(JTH) .GT. CLOCK) GO TO 1005
JTH =VEHNXT(JTH)
GO TO 1001
1005 MOVTIM = VTIME(JTH)
RETURN
1010 MOVTIM = 2**30
RETURN

2 FORMAT(' TERMINATE **I4** AT NODE **I4**)
3 FORMAT(' VEH **I4** PUT IN QUEUE **I4** STATE **I2**)
4 FORMAT(' **I4** TO EXIT QUEUE **I4** AT TIME **I7**
5 FORMAT(' VEH **I4** EXIT QUEUE **I4** TO S1 WILL MOVE **I7**
6 FORMAT(' VEH **I4** LIGHT STOP QUEUE **I4** WILL MOVE **I7**
7 FORMAT(' VEH **I4** BLOCKED AT QUEUE **I4**
8 FORMAT(' VEH **I4** ADVANCED TO NEW LINK **I4**
9 FORMAT(' VEH **I4** EXIT S1 MOVES TO S2 AT NODE **I4**
10 FORMAT(' VEH **I4** EXIT S2 MOVES TO S3 AT NODE **I4**
END
FUNCTION RNKISEED)
**Subroutine Setup**

This subroutine does the final setup on all data files before the simulation begins.

```fortran
C
C This subroutine does the final setup on all data files before the simulation begins.
C
INCLUDE DEFINs
VEGTIM = 2**3n
VEGIND = 0
C
C ..... FOR EACH NODE
DO 2000 NTh=1,NUMNOS
IF(NOTYPE(NTh) .GT. 2) GO TO 1200
IF(NOTYPE(NTh) .EQ. 2) GO TO 2900
C
C ..... GENERATE NODES. SET TIME OF FIRST ARRIVAL.
NOPARS(NTh+1) = 36000. / NOPARS(NTh+1)
DO 1001 I=1,100
1001 A = RNK(ISEED)
NOSEED(NTh) = ISEED
NOCLK(NTh) = TBAGFN(NTh)
C
C ..... FIND LINK TO PUT VEHICLES ON.
DO 1100 KTH=1,NUMLKS
IF(N0NEXT(NTh+1) .GT. LKID(KTH,1)) GO TO 1110
1100 CONTINUE
CALL DUMP(NTh,'SETUP ••'DO1100••1)
1110 NOOUTP(NTh) = KTh
C
C ..... PUT ON CHAIN. IF CLOCK IS SMALL, PUT FIRST.
IF(NOCLK(NTh) .GT. VEGTIM) Go To 1210
VEGTIM = NOCLK(NTh)
NOGNXT(NTh) = VEGIND
VEGIN = NTh
GO TO 2000
C
C ..... PUT ON CHAIN BETWEEN TWO NODES.
1210 NTHLSP = VEGIND
1220 NTHP = NOGNXT(NTHLSP)
IF(NTHP .EQ. NTh) GO TO 1240
IF(NOCLK(NTH) .LE. NOCLK(NTHP)) GO TO 1240
NTHLSP = NTHP
GO TO 1220
1240 NOGNXT(NTh) = NTHP
NOGNXT(NTHLSP) = NTh
GO TO 2000
C
```
C INTERNAL NODES

1500 CONTINUE
ITH = NOSIG(NTH)
IF(ITH .EQ. 0) GO TO 2000
IDNTH = NOIDNO(NTH)

C TAKE EACH LINK AND SEE IF IT IS CONNECTED TO THIS NODE.

DO 1600 KTH = 1, NUMLKS

C CHECK FOR INPUT TO THIS NODE.
IF(LKID(KTH,2) .NE. IDNTH) GO TO 1600

C OK, NOW FIND OUT WHERE IT CAME FROM.
DO 1520 1 = 1, 4
IF(LKID(KTH,1) .NE. NONXT(NTH,1)) GO TO 1525
1520 CONTINUE
CALL DUMP(KTH,'SETUP ') 'DO1520',1
1525 SIGNP(ITH,1) = KTH
1600 CONTINUE
2000 CONTINUE

C FOR EACH LINK FIND THE DESTINATIONS.
DO 3000 KTH = 1, NUMLKS
DO 2500 1 = 1, 3
IF(LKDEST(KTH,1) .EQ. 0) GO TO 2500
DO 2400 KTHCHK = 1, NUMLKS
IF(LKID(KTHCHK,2) .NE. LKDEST(KTH,1)) GO TO 2400
IF(LKID(KTHCHK,1) .NE. LKID(KTH,2)) GO TO 2400
LKDEST(KTH,1) = KTHCHK
GO TO 2500
2400 CONTINUE
2500 CONTINUE
3000 CONTINUE
RETURN
END

SUBROUTINE SIGCHK

C THIS SUBROUTINE UPDATES A SIGNAL LIGHT. ADDITIONALLY,
C THE OTHER LIGHTS ARE CHECKED FOR A STATE CHANGE AT
C THE CURRENT CLOCK TIME.

INCLUDE DEFINS

C FIRST ESTABLISH MAJOR OR MINOR AND IF CHANGE IS REALLY
WANTED NOW.

10 ITH = ABS(SIGIND)
 I = 1
IF(SIGIND .LE. 0) I = 2
IF(SIGCLK(ITH,1) .GT. CLOCK) GO TO 700

C BRANCH BASED ON OLD STATE.
IF(SIGSTA(ITH,1) .GE. 100) 200, 300
OLD STATE WAS GREEN, CHANGE TO AMBER.

100 SIGSTA(ITH*I) = 1
SIGCLK(ITH*I) = SIGCLK(ITH*I) + AMBERT
GO TO 500

OLD STATE WAS AMBER, CHANGE TO RED:

200 SIGSTA(ITH*I) = 2
SIGCLK(ITH*I) = SIGCLK(ITH*I) + SIGRED(ITH*I)
GO TO 500

OLD STATE WAS RED, CHANGE TO GREEN:

300 SIGSTA(ITH*I) = 0
SIGCLK(ITH*I) = SIGCLK(ITH*I) + SIGGRN(ITH*I)

FIND THE NEXT SIGNAL TO CHANGE AND THEN PUT THAT SIGNAL ON THE CHAIN.

500 SIGIND = SIGXT(ITH*I)

ITHOLD = ARS(SIGIND)
IOLD = 1
IF(SIGIND .LE. 0) IOLD = 2

IF(SIGCLK(ITH*I) .GT. SIGCLK(ITHOLD*IOLD)) GO TO 525

PUT BACK AT HEAD OF CHAIN:

SIGIND = ITH
IF(I .EQ. 2) SIGIND = -ITH
GO TO 700

BRACKET A POSITION ON THE CHAIN - BETWEEN AN OLD AND A NEW:

525 ITHNEW = SIGXT(ITHOLD*IOLD)
IF(ITHNEW) 550 530 535
530 INEW = 1
GO TO 575
535 INEW = 1
GO TO 555
550 ITHNEW = -ITHNEW
INEW = 2

IF TIME FOR NEW BIGGER, HAVE FOUND THE SPOT

555 IF(SIGCLK(ITHNEW*INEW) .GE. SIGCLK(ITH*I)) GO TO 575

CHANGE NEW TO OLD AND CHECK NEXT SPOT:

ITHOLD = ITHNEW
IOLD = INEW
GO TO 525
C \begin{verbatim}
      575 IF(I .EQ. 1) GO TO 576
      SIGNXT(IHOLD, IOLD) = -ITH
      GO TO 580
      576 SIGNXT(IHOLD, IOLD) = ITH
      580 IF(INEW .EQ. 1) GO TO 581
      SIGNXT(IH+1) = -ITHNEW
      GO TO 10
      581 SIGNXT(IH+1) = ITHNEW
      GO TO 10
C
      **** LAST SIGNAL TO PROCESS NOW.
C
      700 SIGTIM = SIGCLK(IH+1)
      C**** WRITE(6,1) SIGTIM, ITH, ((SIGCLK(K, J), SIGSTA(K, J), J=1, 2), K=1, NUMSIG)
      C***1 NUMSIG)
      1 FORMAT(* SIGCHK SUBROUTINE SIGTIM = *, T, ITH = *,
      1 I5, 'I = ', I2, '** CLK, STA: ', 10(I9, 'I9', 'II'), '8X*10(I9, 'II'))
      RETURN
      END
      FUNCTION SPDDIS(I, SEED)
      DIMENSION ARY(4)
      DATA ARY / -1.0, 0.07, 0.97, 1.0 /
      X=RN1(SEED)
      DO 100 I=2, 4
      IF(ARY(I) .GE. X) GO TO 200
      100 CONTINUE
      200 SPDDIS = I * 18.0
      RETURN
      END
      SUBROUTINE STAT(K)
C
      C \begin{verbatim}
      C ...... THIS SUBROUTINE CLEARS THE STATISTICS AFTER WARMUP AND THEN PRINTS THEM AFTER THE FINISH.
C
      INCLUDE DEFINS
      C
      C IF FINISH, GO TO 1000
      IF(K .EQ. 2) GO TO 1000
      DO 200 KTH = 1, NUMLKS
      DO 100 I=1, 2
      LKMAXQ(KTH, I) = LQ(KTH, I)
      LKSTOP(KTH, I) = 0
      LKUEL(KTH, I) = 0
      100 LKVL(KTH, I) = 0
      DO 200 I=1, 3
      LKDELD(KTH, I) = 0
      200 LKVD(KTH, I) = 0
      RETURN
C
      C ***** PRINT THE STATISTICS IN THIS SECTION.
C
      1000 TOTIM = FINISH - WARMUP
      WRITE(6, 1)
      1 FORMAT(I9, 'VEHICLE COUNT', 15X, 'VEHICLE VOLUME', 21X,
      1 'VEHICLE DELAY', 14X, 'DELAY PFR VEHICLE', '8X*10 LINK LANES '
      
      \end{verbatim}
\end{verbatim}
DO 1100 KTH=1,NUMLKS
LKVTOT = LKVL(KTH,1) + LKVL(KTH,2)
LKVL1 = LKVL(KTH,1) * 3600 / TOTIM
LKVL2 = LKVL(KTH,2) * 3600 / TOTIM
LKVTOT = LKVL1 + LKVL2
LKDELT = LKDEL(KTH,1) + LKDEL(KTH,2)
DELVL1 = LKDEL(KTH,1) / FLOAT(LKVL(KTH,1))
DELVL2 = LKDEL(KTH,2) / FLOAT(LKVL(KTH,2))
DELVL3 = LKDEL(KTH,3) / FLOAT(LKVL(KTH,3))

1100 WRITE(6,9) KTH,LKVL1,LKVL2,LKVTOT,LKDELT,DELVL1,DELVL2,DELVL3
WRITE(6,9)
9 FORMAT(7X,4X,12X,9X,3F9.2)

DO 1200 KTH=1,NUMLKS
LKSTOT = LKSTOP(KTH,1) + LKSTOP(KTH,2)
DELVL1 = LKDEL(KTH,1) / LKVD(KTH,1)
DELVL2 = LKDEL(KTH,2) / LKVD(KTH,2)
DELVL3 = LKDEL(KTH,3) / LKVD(KTH,3)

1200 WRITE(6,12) KTH,LKSTOP(KTH,1),LKSTOP(KTH,2),LKSTOT,LKMAXQ(KTH,1),LKMAXQ(KTH,2),LKDELT,DELVL1,DELVL2,DELVL3
12 FORMAT(7X,4X,12X,9X,3F9.2)
11 FORMAT(7X,4X,12X,9X,3F9.2)

WRITE(6,18)
18 FORMAT(7X,4X,12X,9X,3F9.2)
19 FORMAT(7X,4X,12X,9X,3F9.2)

WRITE(6,20)
20 FORMAT(7X,4X,12X,9X,3F9.2)
DELVL2 = LKDEL(KTH*2) / FLOAT(LKVL(KTH*2))
DELVL = LKDEL / FLOAT(LKVTOT)
1250 WRITE(6*6) KTH,LKVL(KTH*1),LKVL(KTH*2),LKVTOT,LKMAXQ(KTH*1),
1 LKMAXQ(KTH*2),LKDEL(KTH*1),LKDEL(KTH*2),LKDELT,
2 DELVL1,DELVL2,DELVL
NTOT = NTOT + NINT
NDTOT = NDTOT + NDEL
DEL = NDEL / FLOAT(NINT)
WRITE(6*7) NINT,DFL
1275 CONTINUE
DEL = NDTOT / FLOAT(NTOT)
WRITE(6*8) DEL
RETURN
END

SUBROUTINE STATA(T)

C C ....... PERIOD RY PFRIOD OUTPUT GENERATOR * LKSOUT IS A
C TELLING THE DESIRED OUTPUT LINKS.
C
INCLUDE DEFINES
DIMENSION LKSOUT(U),N1(4),AVG1(4),X1(4),L1(4),LL1(4),LS1(4),LR1(4)
1 ,DEL1(4),NL1(4),NS1(4),NR1(4),DELY1(4)
DATA LKSOUT / 1*3,0*0 /
PARAMETER N=2
GO TO (10*20),I
10 CONTINUE
WRITE(6*4)
DO 15 II=1,N
L1(II)=0
DEL1(II)=0
LL1(II)=0
LS1(II)=0
15 LR1(II)=0
RETURN
20 WRITE(6*1) CLOCK,NUMVEH
DO 30 II=1,N
MM = LKSOUT(II)
L1P = LKVL(MM,1) + LKVL(MM,2)
N1(II)=L1P - L1(II)
L1(II)=L1P
LL1(II)=LKVJO(MM,1) - LL1(II)
L1(II)=LKVJO(MM,1)
NS1(II)=LKVJO(MM,2) - LS1(II)
LS1(II)=LKVJO(MM,2)
NR1(II)=LKVJO(MM,3) - LR1(II)
LR1(II)=LKVJO(MM,3)
IDELP = LKDEL(MM,1) + LKDEL(MM,2)
DELY1(II)=IDELP-DFL(II)
DEL1(II)=IDELP
AVG1(II)=DELY1(II)/N1(II)
X1(II)=DFLP/FLOAT(L1P)
30 WRITE(6*2) L1P,(LKVJO(MM,J)*J=1,3),
1 IDELP,X1(II),N1(II),NL1(II),NS1(II),NR1(II),AVG1(II)
PUNCH 3,CLOCK,NUMVEH,(N1(II),AVG1(II),X1(II)),II=1,N)
RETURN
1 FORMAT (/,,9H CLOCK = *17,13H NUMVEH = *I7)
2 FORMAT(10X*10,18H,F10.2,11H,315,F8.2)
FUNCTION TBAGEN(NTH)

C THIS FUNCTION GENERATES THE TIME BETWEEN ARRIVALS
C AT THE NTH NODE. AN EXPONENTIAL TIME BETWEEN
C ARRIVALS IS ASSUMED.
C
INCLUDE DEFINS

TBAGEN = -NOPARS(NTH) * ALOG(RN1(NoSEED(NTH)))
RETURN

END

SUBROUTINE VCHAIN(JTH, ITIME)

C THIS SUBROUTINE PUTS THE JTH VEHICLE ON THE
C MOVE CHAIN.
C
INCLUDE DEFINS

C IF CHAIN IS EMPTY, PUT AT HEAD.
C IF(MO VFST .NE. 0) GO TO 10
C
MOV FST = JTH
MOVTIM = ITIME
VEHNXT(JTH) = 0
RETURN

C IF TIME FOR FIRST VEHICLE IS LESS THAN ITIME,
C MUST BRACKET A PLACE ON THE CHAIN.
C
10 IF(VTIME(MOVFST) .LT. ITIME) GO TO 20

VEHNXT(JTH) = MOVFST
MOV FST = JTH
MOVTIM = ITIME
RETURN

C BRACKET A SPOT ON THE CHAIN.
C
20 JTHLST = MOVFST
25 JTHP = VEHNXT(JTHLST)

IF(JTHP .EQ. 0) GO TO 30
IF(VTIME(JTHP) .GE. ITIME) GO TO 30
JTHLST = JTHP
GO TO 25

C POSITION FOUND, PUT ON CHAIN.
C
30 VEHNXT(JTHLST) = JTH
VEHNXT(JTH) = JTHP
IF(MOVTIM .GT. ITIME) MOVTIM = ITIME
RETURN

END
SUBROUTINE VEGEN

C C .... THIS SUBROUTINE IS CALLED WHEN IT IS TIME TO CREATE
C C A VEHICLE TO INPUT INTO THE NETWORK. VEGTIM IS THE
C C SCHEDULED ARRIVAL TIME OF THE NEXT UNBLOCKED ARRIVAL.
C C IF VEGFLG NOT EQUAL TO ZERO THEN A VEHICLE IS BLOCKED
C C FROM ENTERING.
C
C INCLUDE DEFINS
C**** WRITE(6,1)
1 FORMAT(' VEGEN SUBROUTINE')
VEGFLG = 0
NTHLST = 0
NTH = VEGIND

C C .... SET UP NEXT NODE TO SEARCH, THEN CHECK CAPACITY OF
C C THIS OUTPUT LINK.
C
100 NTHNXT = NOGNXT(NTH)
KTH = NOOUTP(NTH)
110 IF(LKCAP(KTH) .LE. LKCONT(KTH)) GO To 600

C C .... FIND AN AVAILABLE VEHICLE AND FILL PARAMETERS.
C
LKCONT(KTH) = LKCONT(KTH) + 1
NUMVEH = NUMVEH + 1
IF(NUMVEH .GT. MAXVEH) CALL DlJMPUTH,'VEGEN','MAXVEH',27)
DO 200 JTH=1,MAXVFH
IF(VDSP(JTH) .EQ. 0) GO TO 300
200 CONTINUE
CALL DUMP(0,'VEGEN','MAXVEH',31)
300 VDSP(JTH) = SPDDIS(ISEED)
VSTATE(JTH) = 0
VLK(JTH) = KTH
VLKTIM(JTH) = CLOCK
X = RNK(ISEED)
DO 310 I=1,2
IF(X .LE. LKPROB(KTH,I)) GO TO 320
310 CONTINUE
VTURN(JTH) = 3
VLAN(JTH) = LKLANS(KTH)
GO TO 350

320 VTURN(JTH) = 1
IF(I .EQ. 2) GO TO 330
VLAN(JTH) = 1
GO TO 350

330 VLAN(JTH) = RN1(ISEED) * LKLANS(KTH) + 1
350 ILANE = VLAN(JTH)
VTIME(JTH) = LKLNG(KTH) * 10 / VDSP(JTH) + CLOCK
IF(VTIME(JTH) .LT. LKARVT(KTH,ILANE) + 10) VTIME(JTH) =
1 LKARVT(KTH,ILANE) + 10
VTIME(JTH) = VTIME(JTH) / 10 * 10
LKARVT(KTH,ILANE) = VTIME(JTH)
CALL VCHAIN(JTH,VTIME(JTH))

C C .... SCHEDULE NEXT ARRIVAL AND PUT NODE BACK INTO CHAIN.
C
NOCLK(NTH) = NOCLK(NTH) + TABGFC(NTH)

C**** WRITE(6,2) NTH, JTH, KTH, VTIME(JTH), NOCLK(NTH)
2 FORMAT(' FROM NONE ',I3, ' MOVF VEH ',I4, ' ONTO LINK ',I3,
1     ' NEXT NEW VEH AT ',I8, ' NEXT VEH AT ',F8.2)

C

***** IF THIS WAS LAST ON THE CHAIN, NO CHANGE NECESSARY.
C
IF(NTHNXT .EQ. 0) GO TO 900

C

***** CHECK FOR RELATIVE CHANGE IN ORDER.
C
IF(NOCLK(NTH) .GT. NOCLK(NTHNXT)) GO TO 400
IF(NOCLK(NTH) .LE. CLOCK) GO TO 110
GO TO 900

C

***** MUST FIND NEW PLACE ON CHAIN
C

400 IF(NTHLST .NE. 0) GO TO 410
VEGIND = NTHNXT
GO TO 420

410 NOGNXT(NTHLST) = NTHNXT
420 NTHOLD = NTHNXT
425 NTHNEW = NOGNXT(NTHOLD)
IF(NTHNEW .EQ. 0) GO TO 500
IF(NOCLK(NTHNEW) .GE. NOCLK(NTH)) GO TO 500
NTHOLD = NTHNEW
GO TO 425

C

***** PUT BETWEEN TWO EVENTS OR AT END OF CHAIN.
C

500 NOGNXT(NTHOLD) = NTH
NOGNXT(NTH) = NTHNEW
GO TO 610

C

***** CHECK NEXT NODE.
C

600 NTHLST = NTH
610 IF(NTHNXT .EQ. 0) GO TO 900
IF(NOCLK(NTHNXT) .GT. CLOCK) GO TO 910
NTH = NTHNXT
GO TO 100

C

***** SET FLAG AND FIND NEXT UNBLOCKED ARRIVAL.
C

900 IF(NOCLK(VEGIND) .LE. CLOCK) GO TO 925

C

***** NOTHING DELAYED.
C

VEGTIM = IFIX(NOCLK(VEGIND) + 9.99999999) / 10 * 10
RETURN

C

***** SOME NODE WAS BLOCKED, FIND VEGTIM.
C

925 VEGFLG = 1
NTHNEW = NOGNXT(VEGIND)
950 IF(NOCLK(NTHNEW) .GT. CLOCK) GO TO 975
NTHNEW = NOGNXT(NTHNEW)
IF(NTHNEW GT 0) GO TO 950
VEGTM = 2**30
RETURN
975 VEGTM = IFIX(NOCLK(NTHNEW) + .99999999) / 10 * 10
RETURN
END
## APPENDIX F

### LINK VOLUMES FOR CLOSED NETWORK

<table>
<thead>
<tr>
<th>Light Flow</th>
<th>Medium Flow</th>
<th>Heavy Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>Volume</td>
<td>Link</td>
</tr>
<tr>
<td>1</td>
<td>600.0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>538.5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>550.0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>460.5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>450.0</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>480.6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>250.0</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>254.9</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>250.0</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>322.8</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>250.0</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>304.7</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>250.0</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>294.6</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>250.0</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>267.1</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>500.0</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>422.6</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>500.0</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>466.5</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>600.0</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>499.2</td>
<td>22</td>
</tr>
<tr>
<td>23</td>
<td>250.0</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>227.1</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>250.0</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>293.6</td>
<td>26</td>
</tr>
<tr>
<td>27</td>
<td>250.0</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>293.4</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>250.0</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>302.5</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>250.0</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>287.7</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>563.0</td>
<td>33</td>
</tr>
<tr>
<td>34</td>
<td>539.2</td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td>530.3</td>
<td>35</td>
</tr>
<tr>
<td>36</td>
<td>554.7</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>480.0</td>
<td>37</td>
</tr>
<tr>
<td>38</td>
<td>558.0</td>
<td>38</td>
</tr>
</tbody>
</table>
APPENDIX G

AUTOCORRELATION FUNCTION OUTPUT
SAMPLE AUTOCORRELATION FUNCTION: CONT MODEL - LINK 1

FORMAT SPECIFICATION USED WAS (17X.F6.0)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.36972</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.12793</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.06602</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.02672</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.01423</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.00435</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.00279</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.00181</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.00035</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.00005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.00003</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.00002</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.00001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>0.00000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

224
<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.36972</td>
</tr>
<tr>
<td>2</td>
<td>0.00722</td>
</tr>
<tr>
<td>3</td>
<td>0.03903</td>
</tr>
<tr>
<td>4</td>
<td>0.01244</td>
</tr>
<tr>
<td>5</td>
<td>0.01468</td>
</tr>
<tr>
<td>6</td>
<td>-0.03299</td>
</tr>
<tr>
<td>7</td>
<td>0.00542</td>
</tr>
<tr>
<td>8</td>
<td>0.07241</td>
</tr>
<tr>
<td>9</td>
<td>-0.03255</td>
</tr>
<tr>
<td>10</td>
<td>-0.07199</td>
</tr>
<tr>
<td>11</td>
<td>-0.05903</td>
</tr>
<tr>
<td>12</td>
<td>0.07754</td>
</tr>
<tr>
<td>13</td>
<td>-0.11010</td>
</tr>
<tr>
<td>14</td>
<td>0.12750</td>
</tr>
<tr>
<td>15</td>
<td>-0.01947</td>
</tr>
<tr>
<td>16</td>
<td>-0.02064</td>
</tr>
<tr>
<td>17</td>
<td>-0.04277</td>
</tr>
<tr>
<td>18</td>
<td>-0.04774</td>
</tr>
<tr>
<td>19</td>
<td>-0.11514</td>
</tr>
<tr>
<td>20</td>
<td>0.00052</td>
</tr>
<tr>
<td>21</td>
<td>0.03136</td>
</tr>
<tr>
<td>22</td>
<td>-0.08385</td>
</tr>
<tr>
<td>23</td>
<td>0.05515</td>
</tr>
<tr>
<td>24</td>
<td>0.01636</td>
</tr>
<tr>
<td>25</td>
<td>0.03896</td>
</tr>
</tbody>
</table>
SAMPLE AUTOCORRELATION FUNCTION: CONT MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X, F6.0)

0  1.00000
1  0.07457
2  0.01344
3  0.00594
4  0.11577
5  0.02795
6  0.05947
7  0.04997
8  0.02761
9  0.01895
10 0.04067
11 0.00476
12 0.07639
13 0.05520
14 0.01199
15 0.12389
16 0.04914
17 0.11324
18 0.13799
19 0.03132
20 0.04489
21 0.02791
22 0.00162
23 0.03630
24 0.03384
25 0.05626
26 0.03262
27 0.02093
28 0.01903
29 0.02752
30 0.04580
31 0.08838
32 0.14250
33 0.2204
34 0.02640
35 0.03451
36 0.03513
37 0.02461
38 0.03320
39 0.10868
40 0.08748
41 0.14286
42 0.01801
43 0.11909
44 0.14026
45 0.04347
46 0.02240
47 0.02720
**Partial Autocorrelation Function: Cont Model - Link 5**

<table>
<thead>
<tr>
<th>Lags</th>
<th>Autocorrelation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.07457</td>
</tr>
<tr>
<td>2</td>
<td>0.00795</td>
</tr>
<tr>
<td>3</td>
<td>0.00533</td>
</tr>
<tr>
<td>4</td>
<td>-0.11745</td>
</tr>
<tr>
<td>5</td>
<td>-0.01102</td>
</tr>
<tr>
<td>6</td>
<td>-0.05521</td>
</tr>
<tr>
<td>7</td>
<td>-0.03973</td>
</tr>
<tr>
<td>8</td>
<td>0.02241</td>
</tr>
<tr>
<td>9</td>
<td>0.02551</td>
</tr>
<tr>
<td>10</td>
<td>0.05137</td>
</tr>
<tr>
<td>11</td>
<td>-0.01113</td>
</tr>
<tr>
<td>12</td>
<td>0.07538</td>
</tr>
<tr>
<td>13</td>
<td>0.10665</td>
</tr>
<tr>
<td>14</td>
<td>-0.00861</td>
</tr>
<tr>
<td>15</td>
<td>-0.13104</td>
</tr>
<tr>
<td>16</td>
<td>-0.05345</td>
</tr>
<tr>
<td>17</td>
<td>0.09072</td>
</tr>
<tr>
<td>18</td>
<td>0.17397</td>
</tr>
<tr>
<td>19</td>
<td>-0.03490</td>
</tr>
<tr>
<td>20</td>
<td>0.07790</td>
</tr>
<tr>
<td>21</td>
<td>0.03273</td>
</tr>
<tr>
<td>22</td>
<td>0.07552</td>
</tr>
<tr>
<td>23</td>
<td>0.04057</td>
</tr>
<tr>
<td>24</td>
<td>0.04922</td>
</tr>
<tr>
<td>25</td>
<td>0.00732</td>
</tr>
</tbody>
</table>
### SAMPLE AUTOCORRELATION FUNCTION: UA MODEL - I.NK 1

**FORMAT SPECIFICATION USED WAS (17X.F6.0)**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000</td>
</tr>
<tr>
<td>1</td>
<td>0.29701</td>
</tr>
<tr>
<td>2</td>
<td>0.08609</td>
</tr>
<tr>
<td>3</td>
<td>-0.00494</td>
</tr>
<tr>
<td>4</td>
<td>-0.06654</td>
</tr>
<tr>
<td>5</td>
<td>-0.13899</td>
</tr>
<tr>
<td>6</td>
<td>-0.11765</td>
</tr>
<tr>
<td>7</td>
<td>-0.04743</td>
</tr>
<tr>
<td>8</td>
<td>0.09783</td>
</tr>
<tr>
<td>9</td>
<td>0.05208</td>
</tr>
<tr>
<td>10</td>
<td>-0.05683</td>
</tr>
<tr>
<td>11</td>
<td>-0.08970</td>
</tr>
<tr>
<td>12</td>
<td>0.04169</td>
</tr>
<tr>
<td>13</td>
<td>0.04454</td>
</tr>
<tr>
<td>14</td>
<td>0.10538</td>
</tr>
<tr>
<td>15</td>
<td>0.06677</td>
</tr>
<tr>
<td>16</td>
<td>0.13542</td>
</tr>
<tr>
<td>17</td>
<td>0.08486</td>
</tr>
<tr>
<td>18</td>
<td>-0.03000</td>
</tr>
<tr>
<td>19</td>
<td>-0.18584</td>
</tr>
<tr>
<td>20</td>
<td>-0.15992</td>
</tr>
<tr>
<td>21</td>
<td>-0.00614</td>
</tr>
<tr>
<td>22</td>
<td>-0.00512</td>
</tr>
<tr>
<td>23</td>
<td>-0.01940</td>
</tr>
<tr>
<td>24</td>
<td>0.13428</td>
</tr>
<tr>
<td>25</td>
<td>-0.16509</td>
</tr>
<tr>
<td>26</td>
<td>-0.07061</td>
</tr>
<tr>
<td>27</td>
<td>-0.02900</td>
</tr>
<tr>
<td>28</td>
<td>-0.00397</td>
</tr>
<tr>
<td>29</td>
<td>-0.17634</td>
</tr>
<tr>
<td>30</td>
<td>-0.18913</td>
</tr>
<tr>
<td>31</td>
<td>-0.08268</td>
</tr>
<tr>
<td>32</td>
<td>-0.15089</td>
</tr>
<tr>
<td>33</td>
<td>-0.16152</td>
</tr>
<tr>
<td>34</td>
<td>-0.06014</td>
</tr>
<tr>
<td>35</td>
<td>-0.02350</td>
</tr>
<tr>
<td>36</td>
<td>-0.07494</td>
</tr>
<tr>
<td>37</td>
<td>-0.08995</td>
</tr>
<tr>
<td>38</td>
<td>-0.04731</td>
</tr>
<tr>
<td>39</td>
<td>0.05155</td>
</tr>
<tr>
<td>40</td>
<td>-0.04908</td>
</tr>
<tr>
<td>41</td>
<td>-0.21968</td>
</tr>
<tr>
<td>42</td>
<td>-0.17599</td>
</tr>
<tr>
<td>43</td>
<td>-0.10970</td>
</tr>
<tr>
<td>44</td>
<td>-0.02393</td>
</tr>
<tr>
<td>45</td>
<td>-0.00277</td>
</tr>
<tr>
<td>46</td>
<td>-0.06558</td>
</tr>
<tr>
<td>47</td>
<td>0.03491</td>
</tr>
</tbody>
</table>
PARTIAL AUTOCORRELATION FUNCTION: UB MODEL - LINK 1

1  +.29701
2  -.00241
3  -.03276
4  -.06144
5  -.10998
6  -.04686
7  +.00866
8  +.09655
9  -.01484
10 +.05769
11 +.00210
12 +.03600
13 -.00119
14 +.06830
15 -.04588
16 -.02192
17 +.01510
18 -.09116
19 -.14141
20 +.06680
21 +.03055
22 -.00603
23 +.00538
24 +.10275
25 +.09336
SAMPLE AUTOCORRELATION FUNCTION: UB MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X*F9.0)

<p>|   | 0.00000 | 0.13256 | 0.04789 | 0.01238 | -0.16519 | -0.06050 | -0.04311 | -0.03439 | 0.01300 | -0.05397 | 0.05703 | 0.01026 | -0.06757 | 0.06188 | 0.02620 | 0.07723 | -0.15461 | 0.09451 | -0.13423 | 0.05276 | 0.04784 | 0.01906 | 0.01660 | 0.06637 | 0.07717 | 0.05861 | 0.01602 | 0.01665 | -0.03596 | -0.10957 | -0.06799 | 0.07153 | 0.01717 | 0.03295 | -0.01279 | -0.15391 | -0.0909 | -0.07621 | -0.06678 | 0.05196 | 0.13554 | -0.06632 | 0.02016 | -0.06356 | -0.02208 | 0.08604 | 0.00122 |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.13256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.03085</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-0.02395</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.16530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-0.01814</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-0.01887</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.02724</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-0.03178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-0.04233</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-0.03873</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.01120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-0.10865</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>-0.05954</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.02543</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-0.08784</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>-0.19030</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.11684</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>-0.17406</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>-0.06118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.06631</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.01153</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>-0.08505</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0.03856</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.04264</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>-0.00426</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SAMPLE AUTOCORRELATION FUNCTION: ZONF MODEL - LINK 1

Format specification used: `#A5 (17X, F6.0)`

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
</tr>
<tr>
<td>0.30831</td>
</tr>
<tr>
<td>0.00784</td>
</tr>
<tr>
<td>-0.64982</td>
</tr>
<tr>
<td>-0.00173</td>
</tr>
<tr>
<td>-0.04672</td>
</tr>
<tr>
<td>-0.96364</td>
</tr>
<tr>
<td>-0.09531</td>
</tr>
<tr>
<td>-1.12071</td>
</tr>
<tr>
<td>-0.08773</td>
</tr>
<tr>
<td>-0.04917</td>
</tr>
<tr>
<td>0.06079</td>
</tr>
<tr>
<td>0.06369</td>
</tr>
<tr>
<td>0.02114</td>
</tr>
<tr>
<td>0.19255</td>
</tr>
<tr>
<td>0.02092</td>
</tr>
<tr>
<td>-0.05975</td>
</tr>
<tr>
<td>-0.05344</td>
</tr>
<tr>
<td>-0.01342</td>
</tr>
<tr>
<td>0.01970</td>
</tr>
<tr>
<td>-0.07706</td>
</tr>
<tr>
<td>-0.01708</td>
</tr>
<tr>
<td>0.53173</td>
</tr>
<tr>
<td>0.07570</td>
</tr>
<tr>
<td>0.11443</td>
</tr>
<tr>
<td>0.01284</td>
</tr>
<tr>
<td>0.08320</td>
</tr>
<tr>
<td>0.12423</td>
</tr>
<tr>
<td>0.14314</td>
</tr>
<tr>
<td>0.05181</td>
</tr>
<tr>
<td>-0.11650</td>
</tr>
<tr>
<td>-0.06271</td>
</tr>
<tr>
<td>-0.06921</td>
</tr>
<tr>
<td>-0.03357</td>
</tr>
<tr>
<td>0.06546</td>
</tr>
<tr>
<td>0.07219</td>
</tr>
<tr>
<td>0.00446</td>
</tr>
<tr>
<td>-0.06594</td>
</tr>
<tr>
<td>-0.01401</td>
</tr>
<tr>
<td>0.09910</td>
</tr>
<tr>
<td>0.11320</td>
</tr>
<tr>
<td>0.03027</td>
</tr>
<tr>
<td>-0.05257</td>
</tr>
<tr>
<td>-0.05401</td>
</tr>
<tr>
<td>-0.01443</td>
</tr>
<tr>
<td>0.01604</td>
</tr>
<tr>
<td>-0.07497</td>
</tr>
<tr>
<td>-0.15665</td>
</tr>
</tbody>
</table>
PARTIAL AUTOCORRELATION FUNCTION: ZONE MODEL - LINK 1

1  .30831
2  -0.01581
3  -0.07762
4  .04506
5  -0.05945
6  -0.04512
7  -0.06052
8  -0.08611
9  -0.02903
10 -0.01870
11 -0.05960
12 -0.1142
13 -0.04161
14  .08714
15 -0.03904
16 -0.10233
17  .00059
18 -0.00576
19  .01649
20 -0.00627
21  .04099
22  .07279
23  .03406
24  .07256
25 -0.04710
SAMPLE AUTOCORRELATION FUNCTION: ZONF MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (34X*F6.0)

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000</td>
</tr>
<tr>
<td>1</td>
<td>0.13453</td>
</tr>
<tr>
<td>2</td>
<td>0.01840</td>
</tr>
<tr>
<td>3</td>
<td>0.01984</td>
</tr>
<tr>
<td>4</td>
<td>0.14170</td>
</tr>
<tr>
<td>5</td>
<td>0.02617</td>
</tr>
<tr>
<td>6</td>
<td>0.00473</td>
</tr>
<tr>
<td>7</td>
<td>0.06693</td>
</tr>
<tr>
<td>8</td>
<td>0.06252</td>
</tr>
<tr>
<td>9</td>
<td>0.01944</td>
</tr>
<tr>
<td>10</td>
<td>0.16852</td>
</tr>
<tr>
<td>11</td>
<td>-0.3732</td>
</tr>
<tr>
<td>12</td>
<td>-0.5823</td>
</tr>
<tr>
<td>13</td>
<td>0.2635</td>
</tr>
<tr>
<td>14</td>
<td>0.14025</td>
</tr>
<tr>
<td>15</td>
<td>0.0921</td>
</tr>
<tr>
<td>16</td>
<td>0.02759</td>
</tr>
<tr>
<td>17</td>
<td>-0.3803</td>
</tr>
<tr>
<td>18</td>
<td>-0.1922</td>
</tr>
<tr>
<td>19</td>
<td>0.0139</td>
</tr>
<tr>
<td>20</td>
<td>0.0396</td>
</tr>
<tr>
<td>21</td>
<td>-0.1762</td>
</tr>
<tr>
<td>22</td>
<td>-0.13784</td>
</tr>
<tr>
<td>23</td>
<td>-0.1282</td>
</tr>
<tr>
<td>24</td>
<td>-0.07182</td>
</tr>
<tr>
<td>25</td>
<td>-0.0977</td>
</tr>
<tr>
<td>26</td>
<td>0.0349</td>
</tr>
<tr>
<td>27</td>
<td>-0.0086</td>
</tr>
<tr>
<td>28</td>
<td>-0.0493</td>
</tr>
<tr>
<td>29</td>
<td>0.1052</td>
</tr>
<tr>
<td>30</td>
<td>-0.0451</td>
</tr>
<tr>
<td>31</td>
<td>-0.2991</td>
</tr>
<tr>
<td>32</td>
<td>-0.0260</td>
</tr>
<tr>
<td>33</td>
<td>0.0016</td>
</tr>
<tr>
<td>34</td>
<td>-0.0890</td>
</tr>
<tr>
<td>35</td>
<td>-0.0690</td>
</tr>
<tr>
<td>36</td>
<td>0.07849</td>
</tr>
<tr>
<td>37</td>
<td>0.07238</td>
</tr>
<tr>
<td>38</td>
<td>0.0485</td>
</tr>
<tr>
<td>39</td>
<td>0.16230</td>
</tr>
<tr>
<td>40</td>
<td>0.04761</td>
</tr>
<tr>
<td>41</td>
<td>-0.1368</td>
</tr>
<tr>
<td>42</td>
<td>0.03183</td>
</tr>
<tr>
<td>43</td>
<td>0.07085</td>
</tr>
<tr>
<td>44</td>
<td>0.08076</td>
</tr>
<tr>
<td>45</td>
<td>0.15723</td>
</tr>
<tr>
<td>46</td>
<td>0.0418</td>
</tr>
<tr>
<td>47</td>
<td>0.11659</td>
</tr>
<tr>
<td>Partial Autocorrelation Function: Zone Model - Link 5</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>.13453</td>
</tr>
<tr>
<td>2</td>
<td>.00031</td>
</tr>
<tr>
<td>3</td>
<td>.01685</td>
</tr>
<tr>
<td>4</td>
<td>.13947</td>
</tr>
<tr>
<td>5</td>
<td>-.06502</td>
</tr>
<tr>
<td>6</td>
<td>.00850</td>
</tr>
<tr>
<td>7</td>
<td>-.07224</td>
</tr>
<tr>
<td>8</td>
<td>.00465</td>
</tr>
<tr>
<td>9</td>
<td>.01593</td>
</tr>
<tr>
<td>10</td>
<td>.10438</td>
</tr>
<tr>
<td>11</td>
<td>-.05121</td>
</tr>
<tr>
<td>12</td>
<td>-.07495</td>
</tr>
<tr>
<td>13</td>
<td>.04920</td>
</tr>
<tr>
<td>14</td>
<td>.10509</td>
</tr>
<tr>
<td>15</td>
<td>-.0036</td>
</tr>
<tr>
<td>16</td>
<td>-.04940</td>
</tr>
<tr>
<td>17</td>
<td>-.04197</td>
</tr>
<tr>
<td>18</td>
<td>.08354</td>
</tr>
<tr>
<td>19</td>
<td>-.04761</td>
</tr>
<tr>
<td>20</td>
<td>.07844</td>
</tr>
<tr>
<td>21</td>
<td>-.00942</td>
</tr>
<tr>
<td>22</td>
<td>-.17859</td>
</tr>
<tr>
<td>23</td>
<td>.01428</td>
</tr>
<tr>
<td>24</td>
<td>-.12523</td>
</tr>
<tr>
<td>25</td>
<td>-.04389</td>
</tr>
</tbody>
</table>
### SAMPLE AUTOCORRELATION FUNCTION: NEXT MODEL - LINK 1

**FORMAT SPECIFICATION USED WAS (17X-F6.0)**

<table>
<thead>
<tr>
<th>i</th>
<th>f(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000</td>
</tr>
<tr>
<td>1</td>
<td>0.21530</td>
</tr>
<tr>
<td>2</td>
<td>-0.3266</td>
</tr>
<tr>
<td>3</td>
<td>-0.0714</td>
</tr>
<tr>
<td>4</td>
<td>-0.1799</td>
</tr>
<tr>
<td>5</td>
<td>-0.02757</td>
</tr>
<tr>
<td>6</td>
<td>0.3164</td>
</tr>
<tr>
<td>7</td>
<td>0.2585</td>
</tr>
<tr>
<td>8</td>
<td>0.0456</td>
</tr>
<tr>
<td>9</td>
<td>0.1923</td>
</tr>
<tr>
<td>10</td>
<td>-0.23946</td>
</tr>
<tr>
<td>11</td>
<td>0.05363</td>
</tr>
<tr>
<td>12</td>
<td>0.12656</td>
</tr>
<tr>
<td>13</td>
<td>-0.3959</td>
</tr>
<tr>
<td>14</td>
<td>0.0836</td>
</tr>
<tr>
<td>15</td>
<td>0.05097</td>
</tr>
<tr>
<td>16</td>
<td>-0.0460</td>
</tr>
<tr>
<td>17</td>
<td>0.0709</td>
</tr>
<tr>
<td>18</td>
<td>0.0941</td>
</tr>
<tr>
<td>19</td>
<td>-0.0701</td>
</tr>
<tr>
<td>20</td>
<td>0.07236</td>
</tr>
<tr>
<td>21</td>
<td>-0.07293</td>
</tr>
<tr>
<td>22</td>
<td>-0.11997</td>
</tr>
<tr>
<td>23</td>
<td>-0.07364</td>
</tr>
<tr>
<td>24</td>
<td>0.00539</td>
</tr>
<tr>
<td>25</td>
<td>0.0668</td>
</tr>
<tr>
<td>26</td>
<td>0.06994</td>
</tr>
<tr>
<td>27</td>
<td>0.0731</td>
</tr>
<tr>
<td>28</td>
<td>-0.00741</td>
</tr>
<tr>
<td>29</td>
<td>0.0719</td>
</tr>
<tr>
<td>30</td>
<td>0.0623</td>
</tr>
<tr>
<td>31</td>
<td>-0.02729</td>
</tr>
<tr>
<td>32</td>
<td>0.0422</td>
</tr>
<tr>
<td>33</td>
<td>0.0476</td>
</tr>
<tr>
<td>34</td>
<td>0.05261</td>
</tr>
<tr>
<td>35</td>
<td>-1.15631</td>
</tr>
<tr>
<td>36</td>
<td>-0.05699</td>
</tr>
<tr>
<td>37</td>
<td>0.0755</td>
</tr>
<tr>
<td>38</td>
<td>0.06096</td>
</tr>
<tr>
<td>39</td>
<td>0.0519</td>
</tr>
<tr>
<td>40</td>
<td>0.06071</td>
</tr>
<tr>
<td>41</td>
<td>-0.0793</td>
</tr>
<tr>
<td>42</td>
<td>-1.0287</td>
</tr>
<tr>
<td>43</td>
<td>-1.2202</td>
</tr>
<tr>
<td>44</td>
<td>-0.00024</td>
</tr>
<tr>
<td>45</td>
<td>0.08189</td>
</tr>
<tr>
<td>46</td>
<td>0.05079</td>
</tr>
<tr>
<td>47</td>
<td>0.14182</td>
</tr>
</tbody>
</table>
SAMPLE AUTOCORRELATION FUNCTION: NEXT MODEL - LINK 5

FORMAT SPECIFICATION USED WAS (3/LX,F6.0)

0  1.00000
1  -0.04326
2  -0.3259
3   0.05975
4   0.0879
5   0.08675
6   0.08229
7   0.11556
8   0.0937
9   0.03214
10  0.02019
11  0.01668
12  0.04274
13  0.07115
14  0.04578
15  0.03367
16  0.02914
17  0.02264
18  0.03609
19  0.10175
20  0.02710
21  0.15366
22  0.14273
23  0.11572
24  0.15337
25  0.11167
26  0.06996
27  0.16575
28  0.32151
29  0.04486
30  0.10489
31  0.16577
32  0.11170
33  0.09286
34  0.03373
35  0.03293
36  0.06843
37  0.04085
38  0.07905
39  0.06846
40  0.03491
41  0.15042
42  0.08206
43  0.01285
44  0.08775
45  0.02603
46  0.08679
47  0.02659
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>0.04326</strong></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>0.03452</strong></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>0.05697</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>0.08544</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>0.07650</strong></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>0.07805</strong></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td><strong>0.12694</strong></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><strong>0.09995</strong></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td><strong>0.03760</strong></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>0.02727</strong></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td><strong>0.04646</strong></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td><strong>0.01988</strong></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td><strong>0.03900</strong></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td><strong>0.02524</strong></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td><strong>0.04711</strong></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td><strong>0.01790</strong></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td><strong>0.02016</strong></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td><strong>0.01409</strong></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>0.12082</strong></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td><strong>0.22571</strong></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td><strong>0.14385</strong></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>0.16896</strong></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td><strong>0.21372</strong></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>0.21669</strong></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td><strong>0.02946</strong></td>
<td></td>
</tr>
</tbody>
</table>


43. Wright, P. H., "Simulation of Traffic at a Four-way Stop Intersection," Final Report, Project B-604, School of Civil Engineering, Georgia Institute of Technology, October, 1957.
BIBLIOGRAPHY ON SIMULATION OF TRAFFIC FLOW


Douty, T. R., "GPSS and Traffic Flow Optimization," Proceedings, 8th Annual Conference and Movite Section Meeting, the University of Missouri Engineering Experiment Station, Series No. 65, November, 1966.


Romm, E. D., "Simulation of Traffic at a Two-Way Stop Intersection," School of Civil Engineering, Georgia Institute of Technology, Atlanta, Georgia, September, 1967.


Tolle, J. E., "Digital Computer Simulation of the Car-Following Situation," a working paper on Project EES202 of the Transportation Engineering Center, Department of Civil Engineering, Ohio State University, Columbus, Ohio, 1965.


Wright, P. H., "Simulation of Traffic at a Four-Way Stop Intersection," Final Report, Project B-604, School of Civil Engineering, Georgia Institute of Technology, October, 1967.


Michael Pierce Deisenroth, son of Mr. and Mrs. Clifton E. Deisenroth, was born in Louisville, Kentucky, on April 8, 1944. He was graduated from Chattanooga High School in Chattanooga, Tennessee, in 1962.

He received the degree of Bachelor in Mechanical Engineering from Georgia Institute of Technology in June, 1967. As an undergraduate he received the Combustion Engineering Scholarship Award. During his undergraduate career, he was employed on the Cooperative Program by Sonoco Products Company of Hartsville, South Carolina.

Mr. Deisenroth entered the graduate program in the School of Industrial and Systems Engineering in September, 1967. He completed the requirements for the degree of Master of Science in Industrial Engineering in March, 1971. His thesis was entitled, "Quantitative Utilization of Activity Data for Initial Layouts."

During his graduate program, Mr. Deisenroth served as a teaching and research assistant. In June, 1971, he became an Instructor in the School of Industrial and Systems Engineering.

Mr. Deisenroth married the former Claudia Louise Mathews on June 18, 1966. Their first daughter, Margaret Elaine, was born May 15, 1969. Their second daughter, Michele Leigh, was born June 22, 1972.