

TE
662
.A3
DO.
FHWA-
RD-
76-79

Report No. FHWA-RD - 76-79

DEVELOPMENT AND TESTING OF INTRAS,
A MICROSCOPIC FREEWAY SIMULATION MODEL

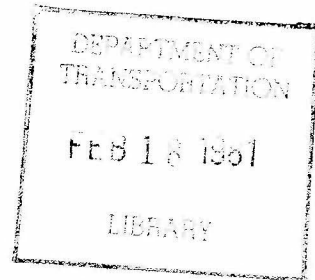
VOL. 4. PROGRAM DOCUMENTATION

D. A. WICKS, B. J. ANDREWS, R. B. GOLDBLATT,



AUGUST 1977

FINAL REPORT



This document is available to the
public through the National Technical
Information Service, Springfield,
Virginia 22151

PREPARED FOR

FEDERAL HIGHWAY ADMINISTRATION
OFFICES OF RESEARCH AND DEVELOPMENT
WASHINGTON, D.C. 20590

Notice

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

The contents of this report reflect the views of KLD Associates, Inc. who is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official view or policy of the Department of Transportation. This report does not constitute a standard, specification or regulation.

662

- 43

FHWA-RD-76-79

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. FHWA-RD-76-79	2. Government Accession No.	3. Recipient's Catalog No.												
4. Title and Subtitle Development and Testing of INTRAS, a Microscopic Freeway Simulation Model Volume 4. Program Documentation		5. Report Date August 1977												
7. Author(s) D. A. Wicks, B. J. Andrews, R. B. Goldblatt,		6. Performing Organization Code												
9. Performing Organization Name and Address KLD Associates, Inc. 300 Broadway Huntington Station, NY 11746		8. Performing Organization Report No. KLD TR-70												
12. Sponsoring Agency Name and Address Office of Research and Development Federal Highway Administration U.S. Department of Transportation Washington, DC 20590		10. Work Unit No. FCP 31C1-042												
15. Supplementary Notes FHWA Contract Manager: S. L. Cohen (HRS-31)		11. Contract or Grant No. DOT-FH-11-8502												
16. Abstract This series of volumes documents the work performed to adapt a freeway simulation model for studying freeway incidents. The resulting program, INTRAS (INTEGRATED TRAFFIC SIMULATION), is a vehicle-specific time-stepping simulation designed to realistically represent traffic and traffic control in a freeway surrounding surface street environment. This volume contains the complete subroutine documentation for the INTRAS work. Each overlay is described along with the formats of any peripheral data files required as input to that overlay. This volume is the fourth in a series. The others in the series are: <table border="1"> <thead> <tr> <th>Vol. No.</th> <th>FHWA No.</th> <th>Short Title</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>76-76</td> <td>Program Design Parameter Calibration and Freeway Dynamics Component Development (Available May 1977)</td> </tr> <tr> <td>2</td> <td>76-77</td> <td>User's Manual (Available June 1977)</td> </tr> <tr> <td>3</td> <td>76-78</td> <td>Validation and Applications (Available August 1977)</td> </tr> </tbody> </table>		Vol. No.	FHWA No.	Short Title	1	76-76	Program Design Parameter Calibration and Freeway Dynamics Component Development (Available May 1977)	2	76-77	User's Manual (Available June 1977)	3	76-78	Validation and Applications (Available August 1977)	13. Type of Report and Period Covered Final Report
Vol. No.	FHWA No.	Short Title												
1	76-76	Program Design Parameter Calibration and Freeway Dynamics Component Development (Available May 1977)												
2	76-77	User's Manual (Available June 1977)												
3	76-78	Validation and Applications (Available August 1977)												
17. Key Words Simulation, Freeway, Freeway Simulation, Traffic Simulation, Freeway Control, Ramp Control, Freeway Operations, Incident Detection		14. Sponsoring Agency Code FHWA												
18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22151		19. Security Classif. (of this report) Unclassified												
20. Security Classif. (of this page) Unclassified		21. No. of Pages 622												
22. Price														

Table of Contents

<u>Section</u>	<u>Title</u>	<u>Page</u>
1	INTRODUCTION	1
2	SUPERVISOR OVERLAY	2
2.1	Program INTRAS	3
2.2	Subroutine DEBUG	5
2.3	Subroutine DRWS	8
2.4	~~Subroutine EROT~~	~~10~~
2.5	Subroutine FINDL	12
2.6	Subroutine FINDLN	15
2.7	Block Data INTVAR	20
2.8	Subroutine LRANK	22
2.9	Subroutine NEXTLN	25
2.10	Subroutine PAK	28
2.11	Subroutine PLNKR	31
2.12	Subroutine PLNKS	34
2.13	Subroutine PVR	37
2.14	Subroutine PVS	40
2.15	Subroutine ULNKR	43
2.16	Subroutine ULNKS	46
2.17	Subroutine UNPAK	49
2.18	Subroutine UVR	53
2.19	Subroutine UVS	56

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
3	PORGIS OVERLAY	59
3.1	Program PORGIS	60
3.2	Subroutine CHKNOD	71
3.3	Subroutine CLRALL	74
3.4	Subroutine CTPFF	77
3.5	Subroutine CTPSV	80
3.6	Subroutine CTPSX	83
3.7	Subroutine DETGEN	86
3.8	Subroutine FLOOUT	89
3.9	Subroutine IMBED	91
3.10	Subroutine IMBEDO	95
3.11	Subroutine INACT	97
3.12	Subroutine INCIN	99
3.13	Subroutine INCOUT	102
3.14	Subroutine INT1	104
3.15	Subroutine LINKIN	108
3.16	Subroutine LINOUT	115
3.17	Subroutine LPAK	118
3.18	Subroutine MATCH	122
3.19	Subroutine PRACT	124
3.20	Subroutine PRMSND	126

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
3.21	Subroutine PRSIG	130
3.22	Subroutine SIGOUT	134
3.23	Subroutine SUROUT	137
3.24	Subroutine SURVIN	139
3.25	Subroutine TABCON	142
3.26	Subroutine TURNIN	144
4	~~LIS OVERLAY~~	~~147~~
5	SIFT OVERLAY	148
5.1	Program SIFT	149
5.2	Subroutine FINDFV	152
5.3	Subroutine FINDRV	154
5.4	Subroutine FINDSV	156
5.5	Subroutine FRSTV	158
5.6	Subroutine LASTLK	161•
5.7	Subroutine LASTV	164
5.8	Subroutine RANDOM	167
5.9	Subroutine VPAK	169
5.10	Program HICON	172
5.11	Subroutine ACTFZ	174
5.12	Subroutine ADVANC	177

91-37

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
5.13	Subroutine ALANE	182
5.14	Subroutine ASIG	184
5.15	Subroutine BLOK	189
5.16	Subroutine CAL1	194
5.17	Subroutine CAL2	197
5.18	Subroutine CAL3	202
5.19	Subroutine CAL4	207
5.20	Subroutine CAL5	214
5.21	Subroutine CANCEL	218
5.22	Subroutine CHANGE	221
5.23	Subroutine CHECK	224
5.24	Subroutine CHOOZ	233
5.25	Subroutine CLNUP	235
5.26	Subroutine CLOSE	239
5.27	Subroutine COLECT	245
5.28	Subroutine CONSOL	249
5.29	Subroutine DECACT	251
5.30	Subroutine DETECT	254
5.31	Subroutine DETSW	257
5.32	Subroutine EMGNCY	259

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
5.33	Subroutine FGNRAT	262
5.34	Subroutine FILL	267
5.35	Subroutine FMAIN	271
5.36	Subroutine FMOVE	274
5.37	Subroutine FRESET	285
5.38	Subroutine FZCL	287
5.39	Subroutine GETCD	289
5.40	Subroutine GETUNV	292
5.41	Subroutine GOQ	294
5.42	Subroutine GRNSIG	300
5.43	Subroutine HDWY	302
5.44	Subroutine INCDAT	304
5.45	Subroutine LANES	308
5.46	Subroutine LCROSS	312
5.47	Subroutine LSWCH	318
5.48	Subroutine MOEV	322
5.49	Subroutine MOOV	325
5.50	Subroutine NORM	335
5.51	Subroutine OFFRMP	337
5.52	Subroutine ONRMP	341
5.53	Subroutine PDAFZ	347

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
5.54	Subroutine PDNAFZ	349
5.55	Subroutine QSTATE	351
5.56	Subroutine REDSIG	353
5.57	Subroutine RELEN	355
5.58	Subroutine REVEN	357
5.59	Subroutine RISK	359
5.60	Subroutine SELEN	362
5.61	Subroutine SEVEN	365
5.62	Subroutine SIGACT	368
5.63	Subroutine SVEH	370
5.64	Subroutine TERM	374
5.65	Subroutine TERMFZ	376
5.66	Subroutine TPTOUT	379
5.67	Subroutine TSIG	382
5.68	Subroutine TSTSAT	386
5.69	Subroutine TYPE	392
5.70	Subroutine UPACT	394
5.71	Subroutine UPSIG	397
5.72	Program LOCON	401
5.73	Subroutine CPTOUT	404
5.74	Subroutine CYCP	406

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
5.75	Subroutine FILTST	414
5.76	Subroutine FTSC	417
5.77	Subroutine INIT	419
5.78	Subroutine INTST	421
5.79	Subroutine RESET	425
5.80	Subroutine SINCL	428
5.81	Subroutine SINC2	432
5.82	Subroutine SINC3	437
5.83	Subroutine SINCES	442
5.84	Subroutine SPOINT	444
6	FUEL OVERLAY	448
6.1	Description of FUEL Data Tape Contents	449
6.2	Program FUEL	450
6.3	Block Data BDCOEM	457
6.4	Block Data BDFLCN	459
6.5	Block Data BDHCEM	461
6.6	Block Data BDNOEM	463
7	INCES OVERLAY	464
7.1	Description of INCES Data Tape Contents	465

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
7.2	Program INCES	467
7.3	Subroutine INCl	476
7.4	Subroutine INC2	480
7.5	Subroutine INC3	485
7.6	Subroutine MOE1	490
7.7	Subroutine MOE2	494
7.8	Subroutine MOE3	498
7.9	Subroutine MOUT	502
7.10	Subroutine PINC	505
7.11	Subroutine POINT	508
7.12	Subroutine POUT	513
7.13	Subroutine READET	516
7.14	Subroutine TTIME	519
8	POSPRO OVERLAY	522
8.1	Program POSPRO	523
9	INPLOT OVERLAY	527
9.1	Description of Plot Data Tape Contents	528
9.2	Program INPLOT	532
9.3	Subroutine AXPLOT	536
9.4	Subroutine COMPCT	539
9.5	Subroutine CONTR	541

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
9.6	Subroutine CPLOT	544
9.7	Subroutine FILEX	548
9.8	Subroutine IOPROC	551
9.9	Subroutine LEGEND	554
9.10	Subroutine PAGE	556
9.11	Subroutine PATH	560
9.12	Subroutine SEARCH	563
9.13	Subroutine SPTAL	565
9.14	Subroutine TPLOT	568
9.15	Subroutine TRAJEC	570
10	SAM OVERLAY	572
10.1	Description of Statistical Data Tape Contents	573
10.2	Program SAM	577
10.3	Subroutine ANOVA	583
10.4	Subroutine CALC	586
10.5	Subroutine PRINT	589
10.6	Subroutine RANK	593
10.7	Subroutine READCL	595
10.8	Subroutine STAT	597

Table of Contents (continued)

<u>Section</u>	<u>Title</u>	<u>Page</u>
10.9	Subroutine TTEST	600
10.10	Subroutine UTEST	603
10.11	Subroutine WILCOX	606
	REFERENCES	609

1. INTRODUCTION

This report, Volume IV of the Final Report for Contract No. DOT-FH-11-8502, contains the subroutine documentation for the INTRAS (INtegrated TRAffic Simula-tion) model. The documentation contained herein is composed of the following:

- Program Narrative - This section describes the subroutine purpose and function. It contains sections on the assumptions made, the approach, a synopsis of the routine and a description of any models or algorithms used.
- Program Variables - This section describes in detail all local variables used within the subroutine. Also described in detail is the subroutine argument list, if any. All variables used which are located in the Common storage area are listed. Descriptions of these variables can be found in the Common Block descriptions in the INTRAS User's Manual.

In addition to the above documentation, complete program listings and machine generated flow charts are available as separate addenda to this report.

The documentation contained in this report is organized as follows. Each major section of this report contains the subroutine documentation for a single overlay of the INTRAS program. The first routine documented in each section is the main program for that overlay. Subsequent subroutines are documented in alphabetical order.

2. SUPERVISOR OVERLAY

The purpose of the supervisor overlay is to sequence the calls to the other functional modules as dictated by the user specified Run Control Cards. This overlay contains subroutines which are required to service routines in a number of other overlays.

2.1 Program INTRAS

Program Narrative

Purpose

INTRAS is the highest-level overlay of the INTRAS simulation. It reads the card type 99 and determines the first overlay to be called, as well as calling subsequent overlays.

Assumptions

None.

Approach

The type 99 card is read and checked for validity. The first overlay call is determined by INTRAS, and subsequent calls to overlays are determined by the last overlay called.

Synopsis

First, various flags and counters are initialized. Then, the next type 99 card is read, and the number of cases processed is incremented. The input parameters are checked: the card type must be 99. The FUEL module parameters must be valid. If the FUEL module is to run with data from tape unit 23 (no simulation will be performed), then that module is called immediately. The case type must be valid. Any old case identification must be valid. The disposition flags, incident run identification specification and incident print flag must all be valid. The INPLOT/SAM flag must be consistent with the case type. If there were any errors, then the run is terminated. If not, then the first overlay call is determined according to the input specifications. Each succeeding overlay is determined by the previous one; a flag is set indicating the next overlay or that the case is over. (See description of /CONTRL/ common block for description of NEXCAL Flag) When the case is over, INTRAS tests the I99LAS flag to see whether more cases follow. If so, then the next case is

processed just as before. If not, then all peripheral units are rewound; FIN is called to terminate plots if the INPLOT module was executed. Finally, the run is terminated.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
ICASNO	I	Counter	Number of cases processed
INEX	I	Switch	Equal to NEXCAL + 1 (see description of NEXCAL), determined branch taken for next overlay

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ ICARD, ICSIN, ICSRN, ICTYPE, IDRUN,
IINCON, IININ, IINOT, INLOC, IPL,
IPT, IPTOT, IRSCT, ISLOT, ISTAT,
ISTOR, I99LAS, LASUBF, NEXCAL

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/CFUEL/ IPTAB, JFUEL, MFUEL, NFLG, NU23

/S2/ ITIME

2.2 Subroutine DBUG

Program Narrative

Purpose

DBGU prints a coded debug print statement.

Assumptions

None.

Approach

Various parameters and dumps are printed depending on the value of NFLAG, the print level flag.

Synopsis

In the default version of DBUG, if the simulation time is less than 999999 (which will always be true), then nothing is printed. (This line of code is to be edited by the user on a given run so that debug print will start at a given simulation time.) If NFLAG is equal to zero, then nothing is printed. If NFLAG is equal to 1 or 2, then the six parameters of the calling sequence are printed. If NFLAG is equal to 1, then nothing more is printed. If the call was not from CLNUP then nothing more is printed. If NFLAG is greater than 1, the simulation time is printed once per time step. If the call was from CLNUP or if NFLAG is equal to 4, a dump is printed of all the link and vehicle array contents. It should be noted that NFLAG is set equal to 4 in a DATA statement but can be changed by a line edit according to the user's needs.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
II	I	Do-Loop Index	Link or vehicle index
III	I	Do-Loop Index	Link or vehicle word being printed
JJ	I	Hold Area	Pointer to last data word for this link or vehicle
JK	I	Do-Loop Index	Set in Do-Loop = (1,2,3,4,5,6) for (freeway, ramp, surface link, freeway, ramp, surface vehicle)
KJ	I	Hold Area	Number of packed words per link or vehicle
KK	I	Hold Area	Offset in number of words to beginning of link or vehicle data
LL	I	Hold Area	Current maximum number of links or vehicles
NFLAG	I	Switch	Print switch, set in data state- ment, equal to (0,1,2,3,4) for (no print, parameter print only, parameter print plus link and vehicle dumps when called by CLNUP plus simulation time, sim- ulation time plus link and ve- hicle dumps when called by CLNUP, simulation time only when called by CLNUP)

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Hold Area	Input, code number indicating where called from
J	I	Hold Area	Input, debug parameter
K	I	Hold Area	Input, debug parameter
L	I	Hold Area	Input, debug parameter
M	I	Hold Area	Input, debug parameter
N	I	Hold Area	Input, debug parameter

Common Blocks and Variables Referenced

/S2/ IDT, ITIME, ITIMEF

/SIZE/ MXLFR, MXLRA, MXLSU, MXVFR, MXVRA,
MXVSL, NLFO, NLRO, NLRO, NVFO, NVRO,
NVSO

/A1/ NLV

2.3 Subroutine DRWS

Program Narrative

Purpose

DRWS performs all the peripheral device operations such as read, write, rewind, backspace, and endfile.

Assumptions

DRWS assumes that control options (see /CONTRL/ common block description) have been set to meaningful values by the calling routine.

Approach

DRWS examines the control options defined by the calling routine and performs the appropriate operation.

Synopsis

First, a branch is taken on the operation flag, IOP: IOP = (1,2,3,4,5) for (read, write, rewind, backspace, endfile). If the operation is to read, then a second branch is taken to read from the specified file with the appropriate format: IFORM = (1,2,3,4,5,6) for (numeric, alpha, type 00 or type 01, type 03, all other formatted cards, binary (unformatted)) card type or record type. In all cases, a flag is set if an end-of-file was encountered while trying to read the record.

Similarly, if the operation is to write, a branch is taken to write a record to the specified file with the appropriate format. IFORM = (1,2,3,4,5,6,7) for (numeric, alpha, numeric list, alpha list, card type write, binary write, buffer dump) record type.

Finally, if either rewind, backspace or endfile is specified, then the operation is performed on the specified file.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-Loop Index	Buffer word index, defined in read or write statement

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IB(2)	I	Hold Area	Input parameter, array of data to be read or written for some formats, dummy argument for other formats and for rewind, backspace, and endfile opera- tions. Note that the dimen- sion (2) is not significant; the length of the actual array passed may be any length, defined by the ILEN variable.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, IFORM, ILEN, IOP

2.4 Subroutine EROT

Program Narrative

Purpose

EROT prints a coded informative error message.

Assumptions

EROT assumes the error code and parameter list have been set meaningfully by the calling program.

Approach

None.

Synopsis

If there is no parameter list, then the error code alone is printed. Otherwise, the error code and parameter list are printed; the user must determine the meaning of the message by looking it up in the user's manual. Finally, the error count is incremented and control returns to the calling program.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>

I	I	Do-Loop Index	Parameter list index, set in write statement from 1 to NPAR

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/ERTRAN/ IEPAR, IER, IERCT, NPAR

2.5 Subroutine FINDL

Program Narrative

Purpose

FINDL finds the link number and link type given the upstream and downstream nodes of a link.

Assumptions

None.

Approach

FINDL searches the SIGI array to find the appropriate link.

Synopsis

FINDL loops over all approaches to the downstream node J in the SIGI array. It unpacks the relative link number and link type and, from these, computes the absolute link number. The corresponding upstream node is found and compared with the given upstream node; if a match is found, then the desired link is found. If none of the approaches match the desired link, an error message is printed (unless it has been suppressed), the link type is returned as zero, and control returns to the calling routine.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Upstream node
ITP	I	Switch	Link type = (1,2,3) for (free-way, ramp, surface) link
J	I	Hold Area	
K	I	Do-Loop Index	Approach number, set in do-loop from 1 to 4
LL1	I	Hold Area	Absolute link number
L1	I	Hold Area	Relative link number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
II	I	Hold Area	Input, upstream node number
IT	I	Switch	Output, link type = (1,2,3) for (freeway, ramp, surface) link, 0 if no link found
JJ	I	Hold Area	Input, downstream node number
L	I	Hold Area	Output, relative link number
LL	I	Hold Area	Output, absolute link number

Common Blocks and Variables Referenced

/A2/ ENDNDS

/CONTRL/ ICARD

/ERTRAN/ IEPAR, IER, NPAR

/S1/ MESERR

/SIZE/ NLFR, NLRA

/A3/ SIGI

2.6 Subroutine FINDLN

Program Narrative

Purpose

FINDLN determines the lane alignment of two freeway links or of a ramp and a freeway link.

Assumptions

FINDLN assumes that the given link number and link types are valid.

Approach

The lane codes for each link are stored, left to right in an array and then shifted so that each receiving lane is aligned with its upstream lane. This array is then returned to the calling program.

Synopsis

First, all arrays and buffers are initialized. Then a branch is taken depending on upstream link type. If the upstream link is a freeway link, then the complete configuration must be determined and appropriate data unpacked: if the downstream link is a ramp only, the number of ramp lanes is unpacked and the ramp is flagged as a right exit ramp (by default). If the downstream links are a freeway exit link and exit ramp combination, then dummy information must be stored for the exit link. The number of ramp lanes must be found, and the ramp is flagged as being a right or left ramp. Otherwise, we must check if an on-ramp is adjacent to the upstream freeway link and, if so, get the number of lanes, the lane in the downstream link receiving its lane 1 traffic, and, hence, whether it is a right or left ramp.

If the upstream link is a ramp only, then we must get the number of ramp lanes. The ramp is flagged as a right ramp (by default). The variable L37 is given a dummy value since it applies only to freeway links (lane receiving lane 1 traffic).

Next, if there is a downstream freeway link, the data for it must be unpacked: the number of lanes, auxiliary lane

codes and auxiliary lane type codes. Also, if an exit ramp is adjacent to this downstream link, then we must get the number of ramp links and whether it is a right- or left-hand ramp.

Then, the lane scheme is determined for the upstream ($K = 1$) and downstream ($K = 2$) links. The array ARRAY is filled with lane codes, the leftmost lane in ARRAY ($K, 1$), etc. Any ramp link lanes are stored as negative numbers to avoid confusion with freeway lanes. A count of the total number of lanes in the upstream and downstream configurations is kept in the LNUM array.

If the upstream link is a ramp only, then the two links are assumed to be aligned with respect to receiving lanes, and processing is finished. Otherwise, lane 1 in the upstream link and lane L37 in the downstream link (receiving lane 1 traffic) must be found and the ARRAY data must be shifted so that these two lanes have the same index. Then all lanes are automatically aligned.

Finally, if the two links have an equal number of lanes, then processing is finished. Otherwise, the extra lanes with no true receiving lane are defined to have as a receiving lane the extreme right or left lane closest to it (if the extra lanes are in the upstream link). Similarly, if the extra lanes are in the downstream link, they are defined as also receiving traffic from the nearest upstream lane.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General Purpose Counter and Do-Loop Index	
ICODE (2)	I	Hold Area	Auxiliary lane type codes
IDIFF	I	Hold Area	No. places data must be shifted to align receiving lanes
IDIFF1	I	Hold Area	11-IDIFF
J	I	General Purpose Do-Loop Index	
K	I	Do-Loop Index	Index to link being processed, = (1,2) for (upstream, downstream) link
KI	I	Hold Area	Index of ARRAY data to be shifted
KK	I	Hold Area	Index of ARRAY location into which data will be shifted
L	I	Switch	(1,2) for (upstream, downstream) link
LC(2)	I	Hold Area	Auxiliary lane codes
LINK (2)	I	Hold Area	Link numbers of upstream and downstream links
LL	I	Switch	(1,2) for (upstream, downstream) link
LNUM (2)	I	Counter	Total number of lanes in upstream, downstream configurations

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LPNT1	I	Hold Area	Index of lane 1 in upstream ARRAY
LPNT2	I	Hold Area	Index of lane receiving upstream lane 1 traffic in ARRAY
LPOS (2)	I	Switch	(0,1) for (right, left) ramp
LR27	I	Hold Area	Lane receiving lane 1 traffic from ramp link
LTEM	I	Hold Area	Exit ramp link number
LTEMP	I	Hold Area	Exit ramp link number
LTYP (2)	I	Hold Area	Link types of upstream and downstream links
L37	I	Hold Area	Lane receiving lane 1 traffic of freeway link (set = -1 for ramp link)
M	I	Hold Area	Number of ramp lanes
MK	I	Hold Area	Index of ARRAY element to be shifted
NK	I	Hold Area	Index of ARRAY location into which data is shifted
NLANE (2)	I	Hold Area	Number of thru lanes in link
NRLANE (2)	I	Hold Area	Number of ramp link lanes

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ARRAY (2,11)	I	Hold Area	Output, array of lane codes, ordered from left to right, such that ARRAY (2,I) is a re- ceiving lane of ARRAY (1,I) in the upstream link
IDLINK	I	Hold Area	Input, downstream link number
IULINK	I	Hold Area	Input, upstream link number
IKTYP	I	Hold Area	Input, link type of upstream link
NKTYP	I	Hold Area	Input, link type of downstream link

Common Blocks and variables Referenced

/SIZE/ MXLRA

2.7 Block Data INTVAR

Program Narrative

Purpose

INTVAR initializes many constants in
labeled common.

Assumptions

None

Approach

None

Synopsis

The constants are initialized as required in
DATA statements.

Models and Algorithms

None

Date, programmer

July, 1975. Dave Wicks.

Program Variables

Local variables

None

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/A6/IANOD
/SIZE/IS, IWDIND, IWDPAK, NLFO, NLFR, NLRA,
NLRO, NLSO, NLSU, NVFO, NVFR, NVRA,
NVRO, NVSO, NVSU, NWDS
/S1/MESERR
/A1/MNLV,
/A4/NDET
/A5/NINCI
/A11/NSTATN
/A2/NTOTL
/A3/NTOTN

2.8 Subroutine LRANK

Program Narrative

Purpose

LRANK establishes a sequence of all freeway links in downstream to upstream order.

Assumptions

None

Approach

LRANK first processes all continuous sequences beginning with entry links and then processes any remaining links not in those sequences. All sequences are stored in the IFRANK array.

Synopsis

First, a list of all freeway entry links is established. LRANK then loops over this list. For each entry link, a sequence is established by examining successive through receiving links until an exit link or no through link is encountered. This sequence is then stored in the IFRANK array, starting in the next available space, in reverse order.

If there are any freeway links remaining which are not included in one of the sequences just established, then a list is made of these links. This list is processed in the same way as the entry links were processed; i.e. sequences beginning with these links are stored, downstream to upstream, in the IFRANK array. The process continues until all freeway links are part of some sequence.

Models and Algorithms

None

Date, programmer

April, 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initilization and Modification</u>
I	I	Counter	Number of entries in LTEMP or IFRANK array
IBEG	I	Hold Area	First location in IFRANK to be filled
IEND	I	Hold Area	Last location of IFRANK to be filled
J	I	Do-loop index	Link index set in do-loop from 1 to NTOTL
K	I	Do-loop index	IFRANK array index, set in do-loop from IBEG to IEND
KK	I	Hold Area	Index to LTEMP entry to be stored
L	I	Do-loop index	Link index, set in do-loop from 1 to MXLFR
LENTY (110)	I	Hold Area	List of freeway entry links
LLEFT (110)	I	Hold Area	List of links not part of sequences processed so far
LNK	I	Hold Area	Link being examined
LTEMP (110)	I	Hold Area	List of links in upstream to downstream order

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
M	I	Hold Area	Number of links in LLEFT list
N	I	Do-loop index	Index to LLEFT and LTEMP arrays, set in do-loop from 1 to m
NEXTL	I	Hold Area	Thru receiving link of link number LNK
NSEQ	I	Hold Area	Number of Sequences in IFRANK array.

Arguments passed to subroutine

None

Common blocks and variables referenced

/A2/ENDNDS, NTOTL

/RANK/IFRANK,NFENT

/SIZE/MXLFR

2.9 Subroutine NEXTLN

Program Narrative

Purpose

NEXTLN determines the downstream link, link type, and receiving lane for a given link, link type (freeway or ramp) and lane number.

Assumptions

NEXTLN assumes that the given link, link type, and lane number are all valid.

Approach

NEXTLN first determines the receiving link and then, with the help of FINDLN, determines which lane is the receiving lane.

Synopsis

First, the receiving link must be found: for a freeway link, the downstream thru movement is examined. If it is an exit link or does not exist (an error condition), the receiving link is set to zero, the link type is set to 2, and processing is over. Otherwise, the through freeway link, or exit ramp link (if it receives all traffic) is found. If any freeway link is adjacent to the ramp link, then it must be used as the upstream link input to FINDLN to avoid confusion.

Next, FINDLN is called to get the lane alignment scheme for the upstream and downstream links. If the upstream link is a ramp link, then the lane number must be made negative for comparison with the result of FINDLN (ramp lane numbers are stored negatively to avoid confusion with freeway lane numbers). The upstream LARRAY location is found which corresponds to the given upstream lane, and then the receiving lane can be found directly. If it is a ramp lane, then a check is made to see if a freeway lane is also fed by the given lane. If so, the freeway lane is returned as the receiving lane, if not, the ramp lane is the result.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Index to LARRAY, lane scheme set in do-loop from II to 11
II	I	Hold Area	Next location in LARRAY after first lane match found
IK	I	Hold Area	Upstream link number
IKTYP	I	Switch	Upstream link type, = (0,1) for (freeway, ramp) link
IL	I	Hold Area	Upstream lane number
K	I	Do-Loop Index	Index to LARRAY, set in do-loop from 1 to 11
LARRAY (2,11)	I	Hold Area	Lane alignment scheme, output from FINDLN, LARRAY(2,K) is re- ceiving lane of LARRAY (1,K) if any
LNEX	I	Hold Area	Thru receiving link
LTEMP	I	Hold Area	Exit ramp number at downstream node of freeway link
NEXL	I	Hold Area	Thru receiving link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIK	I	Hold Area	Input, upstream link number
IIKTYP	I	Switch	Input, upstream link type, (0,1) for (freeway, ramp) link
IIL	I	Hold Area	Input, upstream lane number
LNEXT	I	Hold Area	Output, thru receiving links, if any
LTHRU	I	Hold Area	Output, thru receiving lane, if any
NKTYP	I	Hold Area	Output, receiving link type, (0,1,2) for (freeway, ramp, exit or non-existent) link

Common Blocks and Variables Referenced

/SIZE/ MXLFR

2.10 Subroutine PAK

Program Narrative

Purpose

PAK packs one complete set of parameters for a given link or vehicle into the appropriate array.

Assumptions

PAK assumes the data provided represents valid parameters which can be packed meaningfully.

Approach

PAK loops over all parameters and packs them according to the scheme in the IWDPAK array.

Synopsis

First, for vehicle data, the acceleration and driver type codes are adjusted if necessary. The first and last words of the IWDPAK area to be examined are computed. The offset to the beginning of the NLV area (into which the data will be packed) is computed next. The pointer to the parameter list is initialized, and the first packed data word location and lowest-order digit position are computed.

Then the program loops over all parameters in the parameter list JPARAM. The current packed data word location and lowest-order digit position are saved and those of the next parameter to be processed are computed (to be used the next time through the loop). If the low-order digit position is 6, then it is an artificial flag meaning the packed data word must be made negative if the parameter is equal to 1 (grade sign code) and the packed word can be stored immediately. Otherwise, the packed word is built up by packing the current data word into it. If there are no more parameters to be packed into this word, then it is stored in the NLV array and the NLV pointer is incremented to the next location. Otherwise, processing goes on to the next parameter.

When all parameters have been packed, control returns to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Index to IWDBAK word being processed, set from JJ, MM
IW	I	Hold Area	Packed data word index of parameter currently being processed
IWN	I	Hold Area	Packed data word index of next parameter to be processed
J	I	Counter	Pointer to next parameter to be packed
JJ	I	Hold Area	Location of first IWDBAK word to be processed
JOF	I	Counter	Pointer to NLV word currently being packed
JP	I	Hold Area	Packed data to be stored into NLV
K	I	Switch	Data type, = (1,2,3) for (freeway, ramp, surface) link, (4,5,6) for (freeway, ramp surface) vehicle

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LD	I	Hold Area	Lowest-order digit position of parameter being processed
LDN	I	Hold Area	Lowest-order digit position of next parameter to be processed
MM	I	Hold Area	Last IWDPAK word to be processed
NLVS (6)	I	Constants	Equivalenced to NLFO, etc., offsets to beginning of data for each vehicle type

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IP	I	Switch	Input, link or vehicle type, (1,2,3) for (freeway, ramp, surface) link, (4,5,6) for (freeway, ramp, surface) vehicle
JPARAM (2)	I	Hold Area	Input, array of parameters to be packed
L	V	Hold Area	Input, link or vehicle number

Common Blocks and Variables Referenced

/SIZE/ IS, IWDIND, IWDPAK, NWDS

/A1/ NLV

2.11 Subroutine PLNKR

Program Narrative

Purpose

PLNKR packs one data word into the ramp.
link array.

Assumptions

PLNKR assumes the given data word is within the limits required to pack it meaningfully.

Approach

The given data word is packed in the location defined by the IWDBAK array.

Synopsis

First, the ramp link offset is added to the link number to give the actual location of the link data. The index of the required packed word and the lowest-order digit position within that word of the given parameter are computed. The current packed word is found. Any parameters packed in the high order digits or low order digits aside from the given parameter are unpacked, and then they are repacked with the new value of the given parameter and stored in the LNK array.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold area	Packed data word
IHI	I	Hold area	Parameters packed in higher- order digits beyond given parameter
ILO	I	Hold area	Parameters packed in lower- order digits below given parameter
IP	I	Hold area	Given parameter number
IW	I	Hold area	Index to packed word contain- ing given parameter
IW1	I	Hold area	Index to packed word contain- ing next parameter after given parameter
IWORD (35)	I	Constants	Equivalenced to IWDPAK array, link packing scheme
JP	I	Hold area	Value of given parameter to be packed
L	I	Hold area	Actual location of link data
LD	I	Hold area	Low-order digit position of given parameter
LD1	I	Hold area	Low-order digit position of next parameter after given parameter

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Description, Initialization and Modification</u>
IPl	I	Hold area	Input, parameter number
JPARAM	I	Hold area	Input, parameter value to be packed
LL	I	Hold area	Input, link number

Common blocks and variables referenced

/SIZE/IS, IWDPAK, NLRO
/A1/LNKR

2.12 Subroutine PLNKS

Program Narrative

Purpose

PLNKS packs one data word into the surface link array.

Assumptions

PLNKS assumes the given data word is within the limits required to pack it meaningfully.

Approach

The given data word is packed into the location defined by the IWDBAK array.

Synopsis

First the surface link offset is added to the given link number to get the actual location of the link data. The index of the required packed word and the lowest-order digit position within that word of the given parameter are computed, and the packed word is found. If the parameter to be packed involves only the sign of the packed word (as with grade sign code), then the sign of the packed word is changed if necessary, and processing is finished. Otherwise, the current value of the parameter must be replaced with the new value: any other parameters packed in the high-order digits or low-order digits aside from the given parameter are unpacked and then repacked, preserving the original sign of the packed word. The new packed word is then stored in the surface link array.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed data word
IFAC	I	Hold Area	Sign factor applied to new parameter to preserve original sign of packed word
IHI	I	Hold Area	Parameters packed in higher-order digits beyond given parameter
ILO	I	Hold Area	Parameters packed in lower-order digits below given parameter
IP	I	Hold Area	Given parameter number
IW	I	Hold Area	Index to packed word containing given parameter
IW1	I	Hold Area	Index to packed word containing next parameter after given parameter
IWORD (39)	I	Constants	Equivalenced to IWDPACK array, link packing scheme
JP	I	Hold Area	New value of given parameter to be packed
L	I	Hold Area	Actual location of link data
LD	I	Hold Area	Low-order digit position of given parameter
LD1	I	Hold Area	Low-order digit position of next parameter after given parameter

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IPL	I	Hold Area	Input, parameter number
JPARAM	I	Hold Area	Input, parameter value to be packed
LL	I	Hold Area	Input, link number

Common Blocks and Variables Referenced

/SIZE/ IS, IWDBAK, NLSO

/A1/ LNKS

2.13 Subroutine PVR

Program Narrative

Purpose

PVR packs one data word into the ramp vehicle array.

Assumptions

PVR assumes that the given data word is within the limits required to pack it meaningfully.

Approach

The given data word is packed in the location defined by the IWDPK array.

Synopsis

First, the ramp vehicle offset is added to the vehicle number to give the actual location of the vehicle data . If the given parameter is acceleration/ deceleration, it must be set within the required limits if necessary. If the parameter is the driver type code, it must be decremented by one so that it can be packed meaningfully. The index of the required packed word and the lowest-order digit position within that word of the given parameter are computed. The current packed word is found. Any parameters packed in the high- or low-order digits aside from the given parameter are unpacked, and then they are repacked with the new value of the given parameter and stored in the VR array.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed data word
IHI	I	Hold Area	Parameter packed in higher-order digits beyond given parameter
ILO	I	Hold Area	Parameters packed in lower-order digits below given parameter
IP	I	Hold Area	Given parameter number
IW	I	Hold Area	Index to packed word containing given parameter
IWL	I	Hold Area	Index to packed word containing next parameter after given parameter
IWORD (21)	I	Constants	Equivalenced to IWDPAK array, vehicle packing scheme
JP	I	Hold Area	New value of given parameter to be packed
L	I	Hold Area	Actual location of vehicle data
LD	I	Hold Area	Low-order digit position of given parameter
LDL	I	Hold Area	Low-order digit position of next parameter after given parameter

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IP1	I	Hold Area	Input, parameter number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JPARAM	I	Hold Area	Input, parameter value to be packed
LL	I	Hold Area	Input, vehicle number

Common Blocks and Variables Referenced

/SIZE/ IS, IWDBAK, NVRO

/A1/ VR

2.14 Subroutine PVS

Program Narrative

Purpose

PVS packs one data word into the surface vehicle array.

Assumptions

PVS assumes that the given data word is within the limits required to pack it meaningfully.

Approach

The given word is packed in the location defined by the IWDPACK array.

Synopsis

First, the surface vehicle offset is added to the vehicle number to give the actual location of the vehicle data. If the given parameter is acceleration/deceleration, it must be set within the required limits if necessary. If the parameter is the driver type code, it must be decremented by one so that it can be packed meaningfully. The index of the required packed word and the lowest-order digit position within that word of the given parameter are computed. The current packed word is found. Any parameters packed in the high- or low-order digits aside from the given parameter are unpacked, and then they are repacked with the new value of the given parameter and stored in the VS array.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed data word
IHI	I	Hold Area	Parameter packed in higher order digits beyond given parameter
ILO	I	Hold Area	Parameters packed in lower order digits below given parameter
IP	I	Hold Area	Given parameter number
IW	I	Hold Area	Index to packed word containing given parameter
IWl	I	Hold Area	Index to packed word containing next parameter after given parameter
IWORD (21)	I	Constants	Equivalenced to IWDPAK array, vehicle packing scheme
JP	I	Hold Area	New value of given parameter to be packed
L	I	Hold Area	Actual location of vehicle data
LD	I	Hold Area	Low-order digit position of given parameter
LDl	I	Hold Area	Low-order digit position of next parameter after given parameter

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IPl	I	Hold Area	Input, parameter number
JPARAM	I	Hold Area	Input, parameter value to be packed
LL	I	Hold Area	Input, vehicle number

Common Blocks and Variables Referenced

/SIZE/ IS, IWDPAK, NVSO

/A1/ VS

2.15 Subroutine ULNKR

Program Narrative

Purpose

ULNKR unpacks one data word from the ramp link array.

Assumptions

None

Approach

The desired data word is unpacked and assigned to a formal parameter.

Synopsis

First, the ramp link offset is added to the link number to set the actual location of the link data. Then the packed data word index and the low order digit position of the desired data are computed from the IWORD array (equivalenced to the IWDPK array). The LNKR packed word is found and any digits of lower order than the desired word are discarded. Finally, if there is additional data packed in the high order digits beyond the desired data word, these too are discarded. The desired data word is then passed through the argument list to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, programmer

July, 1975. Dave Wicks

Program variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold area	Packed link array word
IC	I	Hold area	Magnitude of digit beyond highest order digit of desired word
IP	I	Hold area	Parameter number of desired data word
IW	I	Hold area	Index of packed link array word containing desired data
IWORD (39)	I	Constants	Equivalenced to IWDBAK array, link packing scheme
JP	I	Hold area	Desired data word
L	I	Hold area	Actual link index for specified link
LD	I	Hold area	Index to magnitude of low-order digit of desired data
LD1	I	Hold area	Index to magnitude of digit beyond highest order digit of desired data word

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LL	I	Hold area	Input, link number
IP1	I	Hold area	Input, desired parameter number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JPARAM	I	Hold area	Output, desired data word

Common blocks and variables referenced:

/SIZE/IS, IWDBAK, NLRO
/A1/LNKR

2.16 Subroutine ULNKS

Program Narrative

Purpose

ULNKS unpacks one word from the surface link array.

Assumptions

None.

Approach

The desired word is unpacked and assigned to a formal parameter.

Synopsis

First, the surface link offset is added to the given link number to get the actual location of the link data. Then the packed data word index and the low-order digit position of the desired data are computed from the IWORD array (equivalenced to the IWDPAK array). The result, JP, is initialized to zero, and the packed data word is found. If the low-order digit position is equal to 6, then this is an artificial flag (only 5 places are allowed), meaning the sign of the packed data word must be tested: if it is negative, then the result, JP, is set to 1 and control returns to the calling program (this would occur, for example, if the sign of the grade were requested). Otherwise, processing continues as for LNKF: any digits of lower order than the desired word are discarded from the packed data word. If there is any additional data packed in the high order digits beyond the desired data, they, too, are discarded. If the packed word includes a negative sign, then the sign of the result must be changed. The desired data word is then passed through the argument list to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed link array word
IC	I	Hold Area	Magnitude of digit beyond high- est order digit of desired word
IP	I	Hold Area	Parameter number of desired data word
IW	I	Hold Area	Index of packed link array word containing desired data
IWORD (39)	I	Constants	Equivalenced to IWDPAK array, link packing scheme
JP	I	Hold Area	Desired data word
L	I	Hold Area	Actual link index for specified link
LD	I	Hold Area	Index to magnitude of low-order digit of desired data
LD1	I	Hold Area	Index to magnitude of digit beyond highest order digit of desired data word.

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LL	I	Hold Area	Input, link number
IP1	I	Hold Area	Input, desired parameter number
JPARAM	I	Hold Area	Output, desired data word

Common Blocks and Variables Referenced

/SIZE/ IS, IWDBAK, NLSO

/A1/ LNKS

2.17 Subroutine UNPAK

Program Narrative

Purpose

UNPAK unpacks an entire set of parameters for a given link or vehicle number of a given type.

Assumptions

UNPAK assumes that the array passed through the argument list (JPARAM) is of sufficient length to contain all the parameters for that link or vehicle type.

Approach

UNPAK loops over all the packed data for the given link or vehicle and unpacks each data item into the array to be passed through the argument list.

Synopsis

First, the pointers to the first and last words in the IWDPAK array referring to the given link or vehicle type are found. Then the offset, in number of words, to the beginning of the data area for this link or vehicle is computed. Next, the relative word number and low-order digit of the first parameter for this link or vehicle type are computed from IWDPAK. The pointer, J, to the output array, JPARAM, is initialized.

Then UNPAK loops over the NLV array area containing the packed data for the given link or vehicle. The word number and low-order digit for the word now being processed are saved (having been initialized outside the loop) and those for the next word are computed (to be used next time thru the loop). The offset to the word currently being unpacked is computed and the word is found. If the data item is indicated by the sign of the packed data word, then the resulting parameter is set to 0 or 1 depending on the sign. Otherwise, any low-order digits below the desired data are discarded. If there are any higher order digits beyond the desired data word, then they, too, are discarded. If the packed word included a negative sign, then the sign must be

changed. Finally, the result is stored in the next location available in the JPARAM array.

After all parameters have been processed, the driver type code must be incremented by 1 for all vehicle arrays being unpacked. Then the array of unpacked parameters is passed through the argument list to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Index to IWDPACK word being processed, set from JJ, MM packed data word
IBUF	I	Hold Area	Packed data word
IC	I	Hold Area	Magnitude of digit beyond highest order digit of data word
IOF	I	Hold Area	Offset, in no. of words, to beginning of NLV data area
IW	I	Hold Area	Packed data word index, relative to beginning of data area
IWN	I	Hold Area	Packed data word index of next word to be processed
J	I	Counter	Number of unpacked parameters (pointer to next location)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JJ	I	Hold Area	First IWDPACK word to be processed
JOF	I	Hold Area	Offset to NLV word currently being unpacked
JP	I	Hold Area	Unpacked parameter
K	I	Switch	Data type = (1,2,3) for (freeway, ramp, surface) link, = (4,5,6) for (freeway, ramp, surface) vehicle
LD	I	Hold Area	Lowest-order digit position of word being processed
LDN	I	Hold Area	Lowest-order digit position of next word to be processed
MM	I	Hold Area	Last IWDPACK word to be processed
NLVS (6)	I	Constants	Equivalenced to NLFO, etc., offsets to beginning of data for each link and vehicle type

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IP	I	Hold Area	Input, data type, = (1,2,3) for (freeway, ramp, surface) link, = (4,5,6) for (freeway, ramp, surface) vehicle
JPARAM (2)	I	Hold Area	Output, array of unpacked parameters (note that the dimension of JPARAM is not significant; adequate space must be guaranteed by the calling routine)
L	I	Hold Area	Input, link or vehicle number

Common Blocks and Variables Referenced

/SIZE/ IS, IWDIND, IWDPAK, NWDS

/A1/ NLV

2.18 Subroutine UVR

Program Narrative

Purpose

UVR unpacks one word from the ramp vehicle array.

Assumptions

None.

Approach

The desired word is unpacked and assigned to a formal parameter.

Synopsis

First, the ramp vehicle offset is added to the vehicle number to get the actual location of the vehicle data. Then the packed data word index and the low-order digit position of the desired data word are computed from the IWORD array (equivalenced to the IWDPAK array). The VR packed word is found and any digits of lower order than the desired word are discarded. If there is additional data packed in the high order digits beyond the desired data word, then these, too, are discarded. Finally, if the desired data is the driver type code, then the result must be incremented by one (the driver type code has limits of 1 and 10, but packed data must be between 0 and 9). The result is passed through the argument list to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed vehicle array word
IC	I	Hold Area	Magnitude of digit beyond highest order digit of desired word
IP	I	Hold Area	Parameter number of desired data word
IW	I	Hold Area	Index of packed vehicle array word containing desired data word
IWORD (2)	I	Hold Area	Equivalenced to IWDPK array, vehicle packing scheme
JP	I	Hold Area	Desired data word
L	I	Hold Area	Actual vehicle index for specified vehicle
LD	I	Hold Area	Index to magnitude of low-order digit of desired data
LD1	I	Hold Area	Index to magnitude of digit beyond highest order digit of desired data word

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LL	I	Hold Area	Input, vehicle number
IPl	I	Hold Area	Input, desired parameter number
JPARAM	I	Hold Area	Output, desired data word

Common Blocks and Variables Referenced

/SIZE/ IS, IWDPAK, NVRO

/A1/ VR

2.19 Subroutine UVS

Program Narrative

Purpose

UVS unpacks one word from the surface vehicle array.

Assumptions

None.

Approach

The desired word is unpacked and assigned to a formal parameter.

Synopsis

First, the surface vehicle offset is added to the vehicle number to get the actual location of the vehicle data. Then the packed data word index and the low-order digit position of the desired data word are computed from the IWORD array (equivalenced to the IWDPK array). The VS packed word is found and any digits of lower order than the desired word are discarded. If there is additional data packed in the high order digits beyond the desired data word, then these, too, are discarded. Finally, if the desired data is the driver type code, then the result must be incremented by one (the driver type code has limits of 1 and 10, but packed data must be between 0 and 9). The result is passed through the argument list to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

July, 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBUF	I	Hold Area	Packed vehicle array word
IC	I	Hold Area	Magnitude of digit beyond highest order digit of desired word
IP	I	Hold Area	Parameter number of desired data word
IW	I	Hold Area	Index of packed vehicle array word containing desired data word
IWORD (2)	I	Hold Area	Equivalenced to IWDPACK array, vehicle packing scheme
JP	I	Hold Area	Desired data word
L	I	Hold Area	Actual vehicle index for specified vehicle
LD	I	Hold Area	Index to magnitude of low-order digit of desired data
LD1	I	Hold Area	Index to magnitude of digit beyond highest order digit of desired data word

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LL	I	Hold Area	Input, vehicle number
IP1	I	Hold Area	Input, desired parameter number
JPARAM	I	Hold Area	Output, desired data word

Common Blocks and Variables Referenced

/SIZE/ IS,IWDPAK,NVSO

/A1/ VS

3. PORGIS OVERLAY

This overlay examines all simulation oriented input data for errors or inconsistencies and generates printed tables of input data. PORGIS may also store error-free simulation case data on tape.

3.1 Program PORGIS

Program Narrative

Purpose

PORGIS controls the testing of input data for all subintervals of one data case.

Assumptions

None.

Approach

PORGIS checks the type 00, 01, 50, 51, 55, 56, 57 and 60 cards and reports any errors found. Subroutines are called to read the other card types and print input data tables.

Synopsis

The first operation of PORGIS is a test for the production of a table of contents for the case data tape. If the current 99 card requests a table of contents, PORGIS will call TABCON to produce the table. Once TABCON has finished, PORGIS sets NEXCAL to indicate the case is over (which implies the next 99 card should be read) and passes control back to the INTRAS routine.

If the current 99 card is not a table of contents request, PORGIS initializes certain variables and arrays and rewinds file 13 before reading the data for this case. Each data card is copied to file 13 (the file which is read during the checking procedure). In the process, a count is kept of card types 00, 01, 02 and 03 in the ICDCT array. The only error conditions checked for during the copying are an invalid end of input data and a card type greater than type 60. If either of these occur, the case aborts and no more cases will be processed.

Once the case data has been copied to file 13, the checking begins. Tests are performed in a diagnostic loop set to process one subintervals data at a time. For each subintervals data, the same loop is executed. However, not all card types may be submitted after the first subinterval. The presence of a card type not permitted will cause an error message to be output.

In the loop PORGIS checks the number of and contents of the type 00 and 01 cards directly. The processing for each card type is described below:

Card type 00 (Simulation Title Card)

First the number of type 00 cards is checked. ~~(Number is stored in ICDCT(1).) One and only one~~ must be present. The title is stored in the IHEAD array in COMMON/IHEAD/. If the run will produce incident or statistical output tapes, a run identification number must be present. Next, the freeway simulation time-step is checked. If it is not specified, a default of 10 will be used. Otherwise, it must be less than 320 tenths-of-a-second. Also, the production of an incident data tape requires a time-step of 10 tenths-of-a-second. Finally, lane change frequency and random number seed are checked and stored in MAXSTS and RNSEED of COMMON/CONTRL/.

Card type 01 (Network Name Card)

As for the type 00 cards, the number of type 01 cards (from ICDCT(2)) is checked. The network name is stored in the IHEAD array from IHEAD(20) to IHEAD(40). Finally, the various flags and simulation starting time are recorded in the appropriate locations.

Card types 2 through 49 are processed by the following subroutines called by PORGIS:

<u>Subroutine</u>	<u>Card Type Processed</u>
LINKIN	Type 2 through 6
TURNIN	Type 8
PRSIG	Type 10
PRACT	Types 15 through 17
PRMSND	Type 20
SURVIN	Type 25
INCIN	Type 30
IMBED	Types 35 through 49

PORGIS also calls INT1 to test the network geometry. The remaining card types (50,51,55,56,57 and 60) are read and tested directly by PORGIS as follows:

Card Type 50 (Output of Simulation Vehicle Trajectories)

First, the time-step at which vehicle positions will be output is stored (must be at least equal to a time-step). Then the 26 possible nodes describing roadway sections for which data will be output are checked. The nodes must represent freeway links and be in upstream to downstream order with sequences separated by a zero. FINDL is called to determine the link number for each consecutive pair of nodes. The link number and type is then stored in the ITPLNK array.

Card Type 51 (Output of Simulation Contour MOE Data)

First, the time-step at which MOE data will be output is tested. If none is specified, a default of five minutes is used. Next the spacing for contour MOE evaluations is verified; the minimum value is 100 feet. The node sequences are processed as for the type 50 card with the identifying link numbers stored in the ICPLNK array.

Card Type 55 (On-line Incident Detection Specification Card)

On-line incident detection requires the analog/digital mode flag, freeway time-step evaluation frequency, number of parameters to be input and assumed average vehicle length to be set. Also, polling frequency must be set for digital detectors. Error messages will be generated if: a mode flag

other than zero or one is input, a freeway time-step has not been specified, the number of parameters input is less than one or greater than 18 and an assumed average vehicle length is not input.

Card type 56 (On-line Incident Detection Algorithm
Parameter Card)

If a type 55 card was input, at least one type 56 card (two would be necessary if more than nine parameters were specified on the type 55 card) must follow. As each type 56 card is read, the incident detection parameters and algorithm number are stored. The only check made is on the algorithm number which must be 1, 2 or 3.

Card type 57 (On-line Incident Detection Detector
Station Identification Card)

At least one type 57 card must be input if a type 56 was. PORGIS stores the sequence of station numbers in the ISTION array and checks that each station number used appears in only one sequence. The ISTION array is then used to fill the LOCATE, INPOS, KPOINT and NSECTL arrays.

As this point, all card types for one subinterval, except the type 60 card, should have been read. If contour plot data is to be produced during this run, DETGEN is called to generate plot detectors. Then the type 60 card is read. Each field or default parameter is stored in the appropriate common location.

Once the data for the first subinterval has been read, PORGIS will direct the output of data tables by calling the following routines:

<u>Subroutine</u>	<u>Information Output</u>
LINOUT	Link table
SIGOUT	Signal and sign control
FLOOUT	Flow rates
SUROUT	Surveillance data
INCOUT	Incident specifications
IMBEDO	Revisions to imbedded data

Immediately following these tables, PORGIS will print the Control Options Table to identify active options and pertinent parameters.

If it is necessary to read another subintervals data, a branch will be taken to return to the beginning of the diagnostic loop. Otherwise, the next module to be called must be determined. If there have been no errors detected in the case just processed by PORGIS, it will copy the case data to the case data tape, if required, and control will return to the INTRAS module to read the next type 99 card. If errors were found, control will also return to the INTRAS module if no FUEL data cards were input. The presence of FUEL data cards will cause PORGIS to pass control to the FUEL module to process the FUEL data cards.

Models and Algorithms

None.

Date, Programmer

March 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
DIST	I	Hold Area	Distance between two consecutive detector stations
I	I	General Purpose Do-loop Index Switch and Hold Area	
IBLNK	I	Constant	Hollerith constant for a blank field

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDBLZR	I	Hold Area	Counter of number of zeroes after a sequence of station numbers
IDON	I	Hold Area	Contains sum of two consecutive station numbers. IDON=0 indicates end of sequence
IDP	I	Index	Pointer defining number of next detector for this link (index of DTCTR array)
IDPOS	I	Index	Location within DTCTR array that the first detector data for this link is stored
IDPP	I	Index	Pointer defining number of next detector for this link (index of DTCTR array)
IEND	I	Hold Area	Code (0,1) if this (is not, is) the last sensor position
IGOTCD	I	Switch	Flag indicating what position in coding to return to after executing common coding
II	I	General Purpose Hold Area and Index	
II1	I	Hold Area	Last station number stored in ISTION array
II2	I	Hold Area	Current station number to be stored in ISTION array
IK	I	Hold Area	Contents of DTCTR(IDP,2) where IDP is the detector being processed or the unpacked detector type code

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IKP	I	Hold Area	Code (0,1) if this detector (was not, was) generated to output plot parameters
INOTC	I	Hold Area	Code (0,≠0) if either an incident or statistical output tape or both (are not, are) to be produced
INSTA	I	Hold Area	Number of stations whose data is stored in TEMPST array
INTV	I	Hold Area	Incident evaluation period, tenths-of-a-second
IOCARD	I	Hold Area	Type of last card read. Used only while copying types 0 through 3 to file 13
IR	I	Hold Area	Contents of LOCATE array for sensor K
IRR	I	Hold Area	Contents of LOCATE array for sensor KK1
ISAVE	I	Hold Area	Index of the last non-zero entry in ISTION array
ISTATN	I	Hold Area	Station number of detector being processed
ISTION (50)	I	Hold Area	Contains sequences of station numbers as input on type 57 card
IT	I	Hold Area	Link type, 0 indicates a freeway link
ITS	I	Hold Area	Sensor location, feet

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ITSS	I	Hold Area	Station number sensor belongs to
IUP	I	Hold Area	Occupancy updating interval, tenths-of-a-second
IZER	I	Constant	Hollerith constant for zero
IZY	I	Do-loop Index	Station index. Set in Do-loop from 1 to LAZY
J	I	General Purpose Hold Area	
JEND	I	Hold Area	Code (0,1) if this (is not, is) the last sensor position
JSTA	I	Hold Area	Station number corresponding to sensor position KK1
K	I	General Purpose Hold Area and Do-loop Index	
KII	I	Hold Area	Contents of LOCATE array for sensor LK
KK	I	Switch	Code (1,2,3) if card being processed is for (trajectory plots, contour plots, incident detection)
KKT	I	Hold Area	Position in TEMPST array containing data on station currently being processed
KK1	I	General Purpose Hold Area and Do-loop Index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KK2	I	Hold Area	Last position in EPARAM array to be filled during read statement
KNTA	I	Hold Area	Number of stations to be looped over when searching TEMPST array
KSTA	I	Hold Area	Station number being processed
KTPE	I	Hold Area	Detector type
Kl	I	Do-loop Index	TEMPST array index, set in loop from 1 to LNSTA
L	I	General Purpose Hold Area	
LASTCD	I	Hold Area	Type of last card that was read
LASTDT	I	Hold Area	Last detector number at station being processed
LAZY	I	Hold Area	Number of positions in ISTION array to be checked for duplicate stations
LI	I	Hold Area	Link where station KSTA is located
LII	I	Hold Area	Link where station KII is located
LINDEX	I	Hold Area	Index of LNKF array for link station is located on
LK	I	Hold Area	Index of LOCATE array for next station in series
LL	I	General Purpose Hold Area	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LLEN	I	Hold Area	Length of link where station KSTA is located
LLL	I	Dummy Argument to FINDL	
LNEXT	I	Hold Area	Link number of link between stations KSTA and KII
LNSTA	I	Hold Area	Number of positions in TEMPST array to check
MM	I	Hold Area	Link number corresponding to upstream node J and downstream node L
MSTS	I	Hold Area	Freeway simulation time-step
NALG	I	Hold Area	Incident detection algorithm number
NAXSTA	I	Hold Area	Number of positions of LOCATE array to be used in defining NSECTL array
NDIST	I	Hold Area	Average distance of detectors at station KSTA, feet
NMPAR	I	Hold Area	Number of parameters for the incident detection algorithm to be input on card type 56
NUMDT	I	Counter	Count of detectors at station KSTA
TEMPST (50,2)	I	Hold Area	Stations numbers and positions of stations on link LNEXT

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/ LNKF

/A2/ ENDNDS

/A3/ NTOTN

/A4/ DTCTR

/CFUEL/ JFUEL, MFUEL, NU23

/CONTRL/ IBUF, ICARD, ICDCT, ICPLNK,
ICPOP, ICPSP, ICSIN, ICSRN,
ICTYPE, IDMN, IDRUN, IDSEC,
IFILGO, IFN, IFORM, IGO,
IIDLNK, IIDMOD, IIDOP,
IINCON, IININ, IINDT, ILEN,
ILSUB, INLOC, IOP, IPT,
IPTOT, IRSCT, ISFCI, ISTAT,
ISTFLG, ISTOR, ITPLNK,
ITPOP, I99LAS, LASUBF,
MAXSTS, MINSTS, MXINIT, NDT,
NEXCAL, NPRNT, NTRN, RNSEED

/CRASH/ EPARAM, INCALG, INPOS, INTVAL,
IUPDAT, KON, KPOINT, LENTHV,
LNKSTA, LOCATE, MAXSTA,
NSECTL, NUMSTN, TEMPST

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/FORINC/ INCTIM, LINEON

/IHEAD/ IHEAD

/S2/ IHOURS, IMINS, ISECNS, NIDT

/SIZE/ IS, NLFO

3.2 Subroutine CHKNOD

Program Narrative

Purpose

CHKNOD determines whether upstream and downstream nodes read in by LINKIN represent a valid link and, if so, determines the link number.

Assumptions

None

Approach

Various diagnostic checks are made and an error flag is set accordingly to indicate the result to LINKIN.

Synopsis

First, I, the error flag, is cleared to zero. Then the following tests are made and error messages are printed when necessary. IDWN, the downstream node, must be less or equal to NTOTN, the maximum node number allowed. Both IDWN and IUP, the upstream node number, must be greater than zero. If IUP is less than or equal to NTOTN, the remaining tests are skipped and FINDL is called. If not, IUP must be between 700 and 899 inclusive, indicating an entry link. If CHKNOD is called for card type 3, the following tests are skipped. Otherwise it is checked that, if LTYPE is not 3 (indicating a surface link), then IUP must be between 800 and 899. If LTYPE=3 (freeway link) IUP must be between 700 and 799.

When all the tests have been made, FINDL is called to determine the link number. If the link cannot be found, the error flag is set go 1 and control returns to the calling program. If the call was for card type 3, control likewise returns. Otherwise, the link type of the link returned by FINDL is checked to be a freeway, ramp, or surface link for a call to CHKNOD for card types 4, 5, and 6 respectively.

Models and algorithms

None

Date, programmer

July, 1975. Donald Goldman

Program Variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Switch	Link type = (0,1,2) for (freeway, ramp, surface) link, = -1 if link not found
JJ	I	Hold area	Absolute link number

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Defininition, Initialization and Modification</u>
I	I	Switch	Output parameter: (0,1) for (valid, invalid) nodes
IDWN	I	Hold area	Input parameter: downstream node
IUP	I	Hold area	Input parameter: upstream node
LNUM	I	Hold area	Output parameter: relative link number
LTYPE	I	Switch	Output parameter: Link type: = (3,2,1) for (freeway, ramp, surface) link

Common blocks and variables referenced

/CONTRL/ICARD

/ERTRAN/IEPAR, IER, NPAR

/A3/NTOTN

3.3 Subroutine CLRALL

Program Narrative

Purpose

CLRALL initializes various imbedded constants and zeroes out many arrays.

Assumptions

None.

Approach

Each variable is processed as required.

Synopsis

First, current size parameters are cleared. Then all arrays used to store input data are cleared. Imbedded calibration parameters are set next. Finally, various other arrays, counters and flags are cleared for use by the simulation.

Models and Algorithms

None.

Date, Programmer

July 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General purpose do- loop index	
IBLANK	I	Hold area	Set in data statement = hollerith blank

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
J	I	General purpose do- loop index	
JJ	I	Hold area	Points to first location in IPCT to be zeroed out

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

```

/SIZE/ MXLFR, MXLRA, MXLSU, MXVFR, MSVRA,
      MXVSU
/A1/ MNLV, NLV
/A2/ ENDNDS, IVFEET, NTOTL
/A3/ NTOTN, SIGI
/A4/ DTCTR, IDET, NDET
/A5/ INCID, NINCI
/A6/ IACPAR, IANOD, MANOD, MNODE, NPHS,
      SIG, SIGT
/A7/ NS, SNODE
/A8/ CLEN, CSTDCL, FOLK, IALAG, IAMBR,
      IGAP, ILT, IRT, ITGAP, ITPCT, JERK,
      JMPG, LIMSPD, LNMNSP, LSLAG, LSTTIM,
      MAXAC, MINAC SPLPCT, TLNK, VFH, VFHF
/A10/ MNUVR
/A11/ ISTATN
/S2/ ICOINT, ICYCPO, IEXCS, IFDT, IMCYCL,
      ITIME, ITIMER, ITRINT, MCYCL
/P1/ NAMES
/CONTRL/ ICPLNK, ICPOD, ICPSD, IDMN, IDRUN,
      IDSEL, IFILGO, IGO, IIDLNK, IIDOP,
      ILSUB, IODFLG, ISECI, ISTFLG,
      ITPLNK, ITPOP, MAXSTS, MINISTS,
      MXINIT, NDT, NPRNT, NTRN, RNSEED
/FPAR/ INCH, INRAMP, IORD, IP, IRF, ISENS,
      KET, LAG, LAGA, LAGB, MINH DY, MNAC,
      NCH

```

/FRSTVP/ IPOST
/NFILTS/ NDSCHG, NDSCHO, NENTRO, NLASEQ
/A12/ IFCOEF, IGRVAL
/ONVEH/ IORG, IREN, IRV, NORG
/LASDCH/ LADISH
/FORINC/ ICALL, LINEON

3.4 Subroutine CTPFF

Program Narrative

Purpose

To perform diagnostic checks for card type 15. All valid data are stored in the proper actuated array for later reference.

Assumptions

None.

Approach

This routine performs all the necessary diagnostic checks for data that was read on card type 15. Error messages are printed if inconsistency, incompleteness and violation of the network topology are encountered in the data.

Synopsis

The routine first checks whether the actuated node has already been specified. If not, then the actuated node counter is incremented. The routine also makes sure that the maximum number of actuated nodes allowed in the network is not exceeded. The approach links to the actuated control node are stored in NALNK array after they have been verified with the network topology. The red rest code, the controller coordination code and the cycle length are checked for their consistencies. Finally all the above codes and the entry code and the detector switching code are properly stored in their respective array locations.

Diagnostic error messages are printed, if any.

If a diagnostic error is encountered, the routine prints the related error message and increments the error counter,

IERCT. Then, the routine resumes the processing of the remaining data. If, however, the specified actuated node is invalid or exceeded the maximum allowed in the network, the routine will terminate the processing and return to the calling routine after it has printed the proper error message and incremented the error counter.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Hold area	Node specified in input
JJ	I	Hold Area	Downstream node number of approach link
IKK	I	Hold area	Link number
KKK	I	Hold area	Absolute surface link number
KK	I	Hold area	Link specification

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A3/ SIGI

/SIZE/ NLFR, NLRA

/A2/ ENDNDS

/A6/ MNODE

/ERTRAN/ IER, IEPAR

/ACT0/ MACR, NNCR

/ACT7/ NALNK

/ACT8/ NACDS

/ACT9/ NACYC

/ACT10/ NACT

3.5 Subroutine CTPSV

Program Narrative

Purpose

To perform diagnostic checks for card type 17. All valid data are stored in the proper actuated arrays for later reference.

Assumptions

None.

Approach

This routine performs diagnostic checks for phase operation cards, card type 17. It examines those approach links which are specified to be serviced by a given phase and those lanes where detectors are placed, to see whether both inputs are consistent with network topology. Upon completion of processing of a type 17 card, the next type 17 card will be processed. These steps are repeated until all type 17 cards have been read and processed.

Synopsis

Upon entering the routine, it examines the node and the phase specified by this type 17 card to determine if they are indeed a valid actuated node and a valid phase. If not, the related error message is printed and the routine is then ready for the next card. Otherwise, the routine proceeds to verify that all approach links that were specified to be serviced by a given phase are consistent with the network topology. It also makes sure that the lanes where detectors are placed are consistent with the network topology. When processing is finished, the routine reads the next card. It will repeat the above process if the next card is a type 17 card; otherwise, the routine is returned to the caller.

The routine will resume processing after it has printed the related error message and incremented the error counter,

IERCT, when an inconsistency in the network topology is encountered. It will, however, terminate the processing and transfer to read the next card if an invalid actuated node or an invalid phase is encountered.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Hold area	Node
NN	I	Hold area	Actuated node
IPH	I	Hold area	Phase
I	I	Do-loop index	Approach number. Set in do-loop from 1 to 4
KKK	I	Hold area	Stores signals into SIGA array for phase IPH
LL	I	Hold area	Link number where the assumed detector is located
LAN	I	Hold area	Lane number where the assumed detector is located

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A3/ SIGI

/ERTRAN/ IER, NPAR, IEPAR

/CONTRL/ IFN, IOP, IFORM, IBUF

/ACT20/ SIGA

/ACT4/ NADET

/ACT7/ NALNK

/ACT10/ NACT

/ACT12/ NASTAT

3.6 Subroutine CTPSX

Program Narrative

Purpose

To perform diagnostic checks for card type 16. All valid data is stored in the proper actuated array for later reference.

Assumptions

None.

Approach

This routine performs all the necessary diagnostic checks for each actuated or non-actuated phase card type 16. The beginning of yield interval, the end of yield interval and the offset of a non-actuated phase are verified for their relative consistencies with respect to time references. For an actuated phase, the initial interval code, initial interval data and initial actuation data were checked for their validity. All valid phase data are stored in the NAFZ array for later reference.

Synopsis

The routine first checks whether the node specified on this card type is indeed an actuated node. For a non-actuated phase, the relative consistencies with respect to time are verified for those parameters which specify the beginning of the yield interval, the end of the yield interval and the offset.

For an actuated phase, the routine must verify the validity of those specified codes, i.e., initial interval code, IIC, initial interval data, IID, initial actuation data, IAD, recall switching code, memory switch code, inhibit maximum termination code and overlap code. If the actuated controller is a coordinated one, then the phase must have a force-off time specified. A proper combination for the gap reduction parameters must also be verified. All valid phase data are stored in the phase array, NAFZ, for later references.

Diagnostic error messages are printed, if any. If a diagnostic error is encountered, the routine prints the related error message and increments the error counter, IERCT. Then, the routine resumes the processing of the remaining data. If, however, the specified node is not an actuated node, or the specified phase is invalid, or the phase has been specified before, the routine will terminate the processing and return to the calling routine after it has printed the proper error message and incremented the error counter.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Hold area	Node
NN	I	Hold area	Actuated node
IPH	I	Hold area	Phase
I	I	Hold area	Entry number
ICCC	I	Hold area	Coordination code
IID	I	Hold area	Initial interval data
IAD	I	Hold area	Initial actuated data

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/ERTRAN/ IER, IEPAR

/A6/ MNODE

/ACT1/ NAFZ

/ACT3/ NATMR

/ACT8/ NACDS

/ACT9/ NACYC

/ACT10/ NACT

3.7 Subroutine DETGEN

Program Narrative

Purpose

DETGEN creates DTCTR array entries for contour plot detector output data.

Assumptions

DETGEN assumes the user has read in a type 51 card which provides data for the ICPLNK array.

Approach

DETGEN loops over all links which will have detectors and creates the detectors with appropriate parameters.

Synopsis

First, the receiving link array (IRCV) is initialized to zero. Then DETGEN loops over all links in the ICPLNK array. The link number and type are unpacked and then the link data is unpacked from the appropriate link array. The current link is compared with receiving links of the last link processed; (if it is part of a continuous path, then the first detector must be placed on the link a specified distance beyond the last detector on the upstream link). Then the first detector location, ILOC, is compared to link length. If the link is too short, then no detectors can be placed on it. Otherwise, plot detectors are placed in all lanes at this location on the link, giving special consideration to auxiliary lanes and turn pockets. For each lane, a new DTCTR entry must be found. If this is the first detector for this link, then the location must be stored in the link array. If not, then it must be linked to the previous last detector. The detector location, type, plot code, and lane code are all stored.

Then the next detector location is computed. If it is within the link boundary, then the above procedure is repeated. If not, then the location of the first detector on a receiving link, if any, is computed. Finally, the receiving links of the link are saved for comparison with the next link to be processed.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-Loop Index	Index of ICPLNK link being pro- cessed, set in do-loop from 1 to 24
IDET2	I	Hold Area	Pointer to first detector on link
IDN	I	Hold Area	Location of downstream end of auxiliary lane
II	I	Switch	Link type = (1,2,3) for (freeway, ramp, surface) link
IID	I	Hold Area	Pointer to IPACK location of first detector on link
ILOC	I	Hold Area	Location of next detector to be placed on link
IRCV(3)	I	Hold Area	Left, thru, and right receiving links of link being processed
IT	I	Switch	Link type of link in ICPLNK array, = (0,1,2) for (freeway, ramp, surface)
IUP	I	Hold Area	Location of upstream end of auxiliary lane
J	I	Do-Loop Index	Index of IRCV, receiving link array

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JJ	I	Hold Area	Pointer to last detector on link
KK	I	Hold Area	Pointer to next detector on link
L	I	Hold Area	Relative link number of link in ICPLNK array
LCOD	I	Hold Area	Lane number
LEN	I	Hold Area	Link length
LENL	I	Hold Area	Length of auxiliary lane
NA	I	Hold Area	Pointer to auxiliary lane data in IPACK array
NLANE	I	Hold Area	Number of thru lanes on link
NLX	I	Hold Area	Total number of lanes, including auxiliary lanes and turn pockets
NN	I	Do-Loop Index	Index of lane being processed, set in Do-loop from 1, NLX

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/ DTCTR, IDET

/CONTRL/ ICPLNK, ICPSP

/ERTRAN/ IEPAR, IER

/A8/ IPACK

3.8 Subroutine FLOOUT

Program Narrative

Purpose

FLOOUT prints the flow rate data read in by PRMSND.

Assumptions

None.

Approaches

FLOOUT loops through all SNODE entries and prints the data in a table.

Synopsis

First the heading is printed out. Then FLOOUT loops through each SNODE entry. The upstream and downstream nodes are found. Then the vehicle type and lane percentages are unpacked, as well as the flow rates. Then all the data is printed for that entry link.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Entry link index, set in do-loop from 1, NS
ILANE	I	Hold Area	Number of lanes on link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IR	I	Hold Area	Temporary storage for unpacking percentages
J	I	Do-loop Index	Output buffer index
L	I	Hold Area	Absolute entry link
LL	I	Hold Area	Surface entry link
NLFRA	I	Hold Area	Total maximum number of freeway ramp link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/CONTRL/ IBUF, IRSCT, NTRN

/SIZE/ IS, NLFR, NLRA

/A7/ NS, SNODE

3.9 Subroutine IMBED

Program Narrative

Purpose

IMBED reads in and checks diagnostically the imbedded data change cards (types 35-49) and stores the data in the appropriate arrays.

Assumptions

IMBED assumes the card input has been copied to file 13.

Approach

The appropriate action is taken depending on the card type so that the data will be interpreted and stored correctly.

Synopsis

As each card is read, the card type is examined. If it is greater than 49, then the last card read is back-spaced over, and processing is over. If it is less than 35, an error message is printed. Otherwise, it is processed according to type as follows:

Type 35: JMPG index must be valid; if so, data is stored in JMPG. Any non-zero values for ILT, IRT, IALAG are stored. JERK index must be valid; if so, data is stored in JERK.

Type 36: Any elements to be stored in SPLPCT or CLEN arrays must have valid indices.

Type 37: If LSLAG array is specified, elements must be in descending order. ITGAP elements must be ascending order.

Type 38: IAMBR elements, if specified, must be in ascending order. IGAP elements, if specified, must be in descending order.

- Type 39: IGRVAL and MINAC elements, if specified, must have a valid index.
- Type 40: If VFH or VFHF elements are specified, each set must sum to 1000 and be in ascending order.
- Type 41: The first 10 elements of TLNK, if specified, must sum to 1000 and be in descending order. The intersection type index must be valid.
- Type 42: Elements 11 through 20 of TLNK, if specified, must sum to 1000 and be in descending order. The intersection type index must be valid. Elements 21 through 26, if specified are stored.
- Type 43: POLK array element, if specified, must have a valid index.
- Type 44: Each LIMSPD array element, if specified, must have two valid indices.
- Type 45: Each MAXAC element, if specified, must have three valid indices.
- Type 46: Each CSTDCL array element, if specified, must have two valid indices.
- Type 47: Each group of LNMNSP elements, if specified, must have a valid index indicating the number of lanes in question. The group of elements must sum to $100 * \text{number of lanes}$.
- Type 48: Each group of ITPCT elements, if specified, must have a valid index. The group of elements must sum to 100.
- Type 49: Each IFCOEF element, if specified, must have a valid index.

The data is then stored in its appropriate array after having passed its diagnostic tests.

When the first valid imbedded data change card is read, the IMBDIN flag is set to assure that the tables will be printed out.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General purpose Do-Loop Index	
IB	I	Dummy Argument to DRWS	
II	I	General purpose Do-Loop Index	
III	I	Hold Area	Pointer to TLNK location to be primed
IJJ	I	Hold Area	Pointer to IBUF location to be examined
ISUM	I	Hold Area	Sum of Elements
ITOT	I	Hold Area	Value to which ISUM must be equal for LNMNSP or ITPCT
J	I	General Purpose Do-Loop Index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JJ	I	General Purpose Do-Loop Index	
KK	I	Hold Area	Index limit
NTYPE	I	Hold Area	Card Type
JGO	I	Switch	Card Type - 34, indicates branch taken according to type.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A8/ CLEN, CSTDCL, FOLIC, IALAG, IAMBR, IGAP, ILT, IRT, ITGAP, ITPCT, JERK, JMPG, LIMSPD, LNMNSP, LSLAG, MAXAC, MINAC, SPLPCT, TLNK, VFH, VFHF

/CONTRL/ IBUF, IFN, IFORM, ILEN, IOP

/ENTRAN/ IEPAR, IER, NPAR

/A12/ IGRVAL, IFCOEF

/IMBDIN/ IMBDIN

3.10 Subroutine IMBEDO

Program Narrative

Purpose

IMBEDO prints all the imbedded data arrays set in CLRALL or read in by IMBED.

Assumptions

None.

Approach

Each imbedded data array is printed in sequence.

Synopsis

If IMBDIN is not set to 1, then no tables are printed, since no changes were made to the arrays. Otherwise, the variables are printed out in sequence.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General Purpose Do-Loop Index	
J	I	General Purpose Do-Loop Index	
K	I	General Purpose Do-Loop Index	

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A8/ CLEN, CSTDCL, FOLK, IALAG, IAMBR,
IFCOEF, IGAP, ILT, IRT, ITGAP, ITPCT,
JERK, JMPG, LIMSPD, LNMNSP, LSLAG,
MAXAC, MINAC, SPLPCT, TLNK, VFH, VFHF

/A12/ IFCOEF, IGRVAL

/IMBDIN/ IMBDIN

3.11 Subroutine INACT

Program Narrative

Purpose

To print all input specifications for all actuated nodes in the network.

Assumptions

None.

Approach

This routine prints all input specifications for all actuated nodes in the network. For each actuated node, there will be two tables printed. One table contains the locations of the detectors and the phases they service; another table contains detailed specifications of each actuated or non-actuated phase.

Synopsis

This routine prints all input specifications for all actuated nodes in the network. For each actuated node, the routine first prints the coordination code, rest-in-red code, cycle length (if coordinated), entry code, detector switching code. Then, it prints a table to indicate the lane where the detectors are located and the phases for which they are furnishing the actuation calls (for the green and the initial intervals). Finally, a detailed table containing each actuated or non-actuated phase is printed.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A4	R	Hold area	Maximum passage time
A5	R	Hold area	Amount of gap reduction
KW	I	Hold area	Code =(0,1,2)if controller is(single ring, dual ring, dual entry)
KX	I	Hold area	Code = (0,1) if controller (is not, is) coordinated
KY	I	Hold area	Code = (0,1) if controller will (dwell in current ac- tive phase, terminate cur- rent active phase and rest in red)

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/ACT0/ MACR, NNCR, ISTDET

/ACT1/ NAFZ

/ACT4/ NADET

/ACT7/ NALNK

/ACT8/ NACDS

/ACT9/ NACYC

/ACT10/ NACT

/SIZE/ NLRA, NLFR

/A6/ MNODE

3.12 Subroutine INCIN

Program Narrative

Purpose

INCIN reads the incident specification cards (type 30), checks the data diagnostically, and stores the data in the INCID array.

Assumptions

INCIN assumes the card data has been copied to file 13.

Approach

As each card is read, the data is checked diagnostically, and it is then stored in the INCID array.

Synopsis

As each card is read, the end-of-file flag is checked. If an end of file is reached, an error message is printed and processing is over. If not, the card type is checked. If it is less than 30, a message is printed and processing goes on to the next card. If it is greater than 30, then the last card read is backspaced over and processing is over. If it is type 30, then each of two fields on the card is checked. The node numbers must be valid and must correspond to a freeway link defined by the link geometry cards. The link data is unpacked from the link array. Then the incident codes on the input card are unpacked and checked for validity, and each code given must correspond to an existing lane. If an incident is specified for an auxiliary lane but the location is not within the bounds of the lane, the code is reset to 0.

Next, the incident location is checked. It must be within the link boundaries. Then an entry in the INCID array must be found. If this is the first incident for the link, the location must be stored in the link array; if not, it must be linked to the previous last incident. Finally, the data is stored in the INCID array. When both fields have been processed, processing goes on to the next card.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	(1) packed incident code word, set in do-loop from 1,3 (2) Lane number, 1,9
IB	I	Dummy argument to DRWS	
ICOD(9)	I	Hold Area	Equivalenced to IPACK; unpacked incident codes for all lanes.
IDNA	I	Hold Area	Location of downstream end of auxiliary lane
IDNI	I	Hold Area	Location of downstream end of incident
IR	I	Hold Area	Temporary storage used when unpacking incident codes
IT	I	Switch	Link type = (1,3) for (freeway, ramp, surface) link
IUPA	I	Hold Area	Location of upstream end of auxiliary lane
IUPI	I	Hold Area	Location of upstream end of incident
J	I	Do-Loop Index	Input card field index, set in do-loop from 1,2

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JBUF (13,2)	I	Hold Area	Equivalenced to IBUF; input card data
JJ	I	Hold Area	(1) Auxiliary lane data index (2) Pointer to next detector
L	I	Hold Area	Relative link number, computed by FINDL
LDAT (37)	I	Hold Area	Unpacked link data
LL	I	Hold Area	Absolute link number
LOLD	I	Hold Area	Link number of last (previous) incident processed
NCARD	I	Counter	Number of type 30 cards read in
NNC	I	Counter	Number of entries in INCID array

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, ICARD, IFN, IFORM, ILEN, IOP

/ERTRAN/ IEPAR, IER, NPAR

/A5/ INCID, NINCI

/SIZE/ IS

/A3/ NTOTN

3.13 Subroutine INCOUT

Program Narrative

Purpose

INCOUT prints the incident specification data processed by INCIN.

Assumptions

None.

Approach

INCOUT loops over all freeway links and prints out the data for each link.

Synopsis

First, the table heading is printed. Then each freeway link is processed. If there are no incidents, then the link is dismissed from consideration. Otherwise, for each incident on the link, the incident codes for each lane, the incident location, length, start time, duration, and rubberneck factor, if any, are printed.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Freeway link index, set in do-loop from 1, MXLFR

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IPTR	I	Hold Area	Pointer to incident on link being processed
IR	I	Hold Area	Temporary storage for unpacking incident codes
IRUB	I	Hold Area	Rubberneck factor
J	I	General Purpose Do-Loop Index	
JJ	I	Do-Loop Index	Thru or auxiliary lane index, set from 1,5 or 1,4
LL	I	Hold Area	Pointer to next incident on link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/CONTRL/ IBUF

/A5/ INCID

/SIZE/ IS, MXLFR

3.14 Subroutine INT1

Program Narrative

Purpose

INT1 performs the network-wide diagnostic checks on the network defined by LINKIN to assure that the network is meaningful and consistent with certain requirements. Also, some additional link data items are primed at this time.

Assumptions

None.

Approach

INT1 loops over all freeway links, then all ramp links and all surface links, making the appropriate checks for each type. Finally, checks are made on the early warning sign specification.

Synopsis

First, all freeway links are processed (this section is skipped if there are no freeway links): the link data is unpacked and the node numbers are determined. Then a branch is taken to common coding to check that link (IDWN, IUP), if it exists, is the same link type as the link (IUP, IDWN) in question. Next, if an exit ramp is specified as being downstream of the given link, it must have a valid node number. If not an exit node, FINDL must be able to determine the link number. The link type must be 2 (ramp). The node number indicating the exit ramp is replaced with the link number in the freeway link data buffer, and the flag is set for off-ramp in the ramp link array. Next, the thru receiving is checked: the downstream node number of the thru receiving link must be valid. If not an exit node, FINDL is called to determine that the link exists and is a freeway link. The downstream node number is replaced with the link number in the freeway link array buffer. Next, the lane specified in the thru receiving link as receiving lane 1 traffic from the given link must exist. If the thru receiving link is an exit link, then the downstream node must be between 700 and 799. Finally, the data is repacked

into the ramp link array.

Finally, all surface links (if any exist) are processed: the link data is unpacked from the surface link array, and consistency between the surface link (IUP, IDWN) and (IDWN, IUP) is checked by the common coding. The turn-pocket capacity must not exceed that of the entire link. Each of three receiving links is examined: the downstream node specified must be valid, the receiving link must exist and must be a surface or ramp link; if a ramp link, it must be an on-ramp. The downstream receiving node is replaced by the link number in the data buffer. If there are any left-turners on the given link, then the link whose thru traffic opposes left-turners, if any, must be valid: if a ramp link, it must be an off-ramp. It must not be a freeway link. The opposing link number is stored in the link data buffer. Finally, the link data is repacked into the surface link array.

The last section of code checks all freeway links for consistent early warning sign inputs. Both a valid sign location and node referred to must be specified. The node must be the downstream node of a freeway link and the upstream node of an off-ramp link. The off-ramp must be downstream of the link on which the sign appears.

Models and Algorithms

None.

Date, Programmer

July, 1975. Donald Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
BUF (47)	I	Hold Area	Buffer containing unpacked receiving link data, equivalenced to ITPLNK array

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
BUFFER (47)	I	Hold Area	Buffer containing unpacked link data, equivalenced to IPACK array
I	I	General Purpose Do-Loop Index	
IB	I	Hold Area	Downstream node of receiving link
IDWN	I	Hold Area	Downstream node of link
IHOLD	I	Hold Area	Downstream node of thru receiving link
II	I	Hold Area	Node referred to by early warning sign
IRET	I	Switch	Return to statement number (407,507, 607) if common coding entered from (freeway,ramp,surface) code area
IT	I	Switch	(1,2,3) for (freeway, ramp, surface) link, 0 if no link found
ITMP	I	Hold Area	Exit ramp link number
IUP	I	Hold Area	Upstream node of link
J	I	Hold Area	Approach link and link type
JJ	I	Hold Area	Link type of thru receiving link of ramp link
K	I	Hold Area	Pointer to receiving link being checked
KK	I	Switch	(1,2,3) if (freeway, ramp, surface) links being processed
L	I	Hold Area	Link number
LCAP	I	Hold Area	Turn-pocket capacity (number of cars)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>

LNUM	I	Hold Area	Link number of opposing link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/CONTRL/ ICPLNK, ICPOP, ITPLNK

/ERTRAN/ IEPAR, IER, NPAR

/A8/ IPACK

/SIZE/ IS, MXLFR, MXLRA, MXLSU, NLFR, NLRA

/S1/ MESERR

/A3/ NTOTN, SIGI

3.15 Subroutine LINKIN

Program Narrative

Purpose

LINKIN reads in and checks diagnostically the link geometry cards, types 2-6, and primes the appropriate arrays.

Assumptions

LINKIN assumes the input data has been copied to file 13.

Approach

As each card is read, the card type is determined. If valid, the appropriate branch is taken so that the data may be interpreted, checked, and stored, and the appropriate error messages printed out.

Synopsis

For the first subinterval, arrays and counters must be zeroed out. DRWS flags must be set. Then, as each card is read, it is checked as follows: If the card type is greater than 6, then that record is backspaced over and final cleanup operations are performed (to be described later). If it is not the first subinterval and the card type is less than 5, an error message is printed and the card is ignored. For the first subinterval, the card types must be input in ascending order. If the card type is valid, a branch is taken according to card type, the data is checked and stored, and then the next card is processed in the same way until all card type 2-6 have been processed. The processing for each card type is described below:

Card type 2 (link geometry)

First, the data as read from the input card is unpacked so that each data item is stored in a separate buffer word. As each link is processed, a count is kept of the total number of links so that the maximum is not exceeded. The link type must be valid; if not specified, it is assumed to be a freeway link. The nodes specifying the receiving links

must be valid; if they are between 700 and 899, then MATCH must be called to check for uniqueness of the low-order digits.

Then, a branch is taken according to link type. For surface links, the number of surface links is incremented. If necessary, LPAK is called to reallocate link space to allow more surface links. The absolute link number (index to ENDNDS) is computed. The indices for free-flow speed and grade parameters are saved. Receiving node numbers, which will later be changed to link numbers by INT1, are stored in BUFFER, the array which will contain link data to be packed.

Similarly, for ramp links, a count is kept of the number of ramp links, and LPAK is called if necessary. Free-flow speed and grade indices are saved. The movements specified are checked: at least one must be specified, including at least one through movement.

Finally, for freeway links, the count must be incremented, and LPAK is called if necessary. Link length is divided by three to allow for longer freeway links than would otherwise be possible. (A default length of 3275 must be set for freeway entry links). The free-flow speed and grade indices are saved. The early warning sign data, if any, is stored. One downstream through link must be specified, plus at most one other movement to an off-ramp.

Processing continues for all link types: the upstream and downstream nodes must be valid, and MATCH must be called for any entry link upstream nodes. The node numbers are stored in the ENDNDS array. Then, for links whose downstream node is not an exit node, the SIGI array, containing the approach links for each node, must be primed.

Next, the link length and mean desired free-flow speed must be valid. The number of thru lanes must be no more than five. The grade index and grade sign must be computed from the input grade value. All these items are stored in the appropriate buffer locations depending on link type.

Finally, PAK is called to pack the BUFFER data into the appropriate link array. Any remaining fields on the card are processed in the same way, and the next card is read.

Card Type 3 (Link Names)

Each of the three possible fields is checked as follows: the upstream and downstream nodes are read from the input buffer, and CHKNOD is called to check them for validity. If valid, then CHKNOD returns the relative link number. From this the absolute link number is computed, and the hollerith link name is stored in the NAMES array.

Card Type 4 (Freeway link operation)

Each of two fields per card is processed as follows: first, the data as read from the input card is unpacked so that each data item is stored in a separate buffer word. CHKNOD is called to check for valid upstream and downstream nodes and to determine the corresponding link number. UNPAK is called to unpack the link array vector so that the incoming data can be added to it. The radius of curvature, pavement code, auxiliary lane codes and auxiliary lane lengths must all be valid. Then the lane in the through receiving link which receives lane 1 traffic from this link is stored, as well as any specification of lane barriers. PAK is then called to repack the updated link vector. Finally, if any statistical data stations were specified, the location is checked, and the information is stored in the LSTATN array.

Card Type 5 (Ramp link operation)

Each of two fields per card is processed as follows: first, the data as read from the input card is unpacked so that each data item is stored in a separate buffer word. CHKNOD is called to check for valid upstream and downstream nodes and to determine the corresponding link number. If it is the first subinterval, the pavement code is checked for validity and stored. The downstream intersection type must be valid. The mean queue discharge headway is stored. The remaining items specified in the first subinterval only are stored. Finally, PAK is called to repack the link data into the LNKR array.

Card Type 6 (Surface link operation)

Each of two possible fields per card is processed as follows: first, the data as read from the input card is unpacked so that each data item is stored in an individual

buffer word. CHKNOD is called to check for valid upstream and downstream nodes and to determine the corresponding link number. Then the downstream intersection type, mean queue discharge headway, and first vehicle lost time are all checked for validity and stored. If it is the first subinterval, the upstream node of the link whose thru traffic opposes left-turners from this link, as well as the right and left turn pockets, if any, are stored. (These items must be checked for validity by INT1.) If these items are specified after the first subinterval, a warning message is printed. Next, the total number of lanes, including pockets, must not exceed the maximum allowed. Channelization codes must be valid. Finally, PAK is called to repack the link data into the LNKS array.

When all cards type 2-6 have been read, the following cleanup operations are performed: LPAK is called to compress the link arrays, i.e., reset the pointers to remove any unused space allocated to the link arrays so that it may be used later by the vehicle arrays. If it is the first subinterval, then the CHEK78 array (see description of MATCH subroutine) is scanned and messages are printed noting any error conditions. Finally, control returns to the calling program.

Models and Algorithms

None.

Date, Programmer

July, 1975. Donald Goldman

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
BUFFER (47)	I	Hold Area	Link data to be packed into link link array; equivalenced to IPACK
I	I	General Purpose Do-Loop Index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IB(2)	I	Dummy Argument to DRWS	
IBUF2 (13,2)	I	Hold Area	Input card data by field, equivalenced to IBUF
IBUF3 (8,3)	I	Hold Area	Input card data by field, equivalenced to IBUF
IBUF5 (13,2)	I	Hold Area	Input card data by field, equivalenced to IBUF
ICODE	I	Hold Area	Grade code sign = (0,1) for (positive, negative) grade
ICOUNT	I	Counter	Total number of links of all types
IDIF	I	Hold Area	Difference between two consecutive calibrated grade values
IDWN	I	Hold Area	Downstream node of link
IGI	I	Hold Area	Grade index
IGR	I	Hold Area	Grade value on input card
IGRADE	I	Hold Area	Grade index location in BUFFER array
IR	I	Hold Area	Temporary storage for unpacking input data
IT	I	Hold Area	Temporary storage for unpacking input data
ITMP	I	Hold Area	Link length
IUP	I	Hold Area	Upstream node of link
J	I	General Purpose Do-Loop Index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JCD (2)	I	Counter	Number of cards type 2 and 3 read in so far
JGRADE	I	Hold Area	Grade sign location in BUFFER array
K	I	Do-Loop Index	Field number on input card being processed
LAUXC	I	Hold Area	Auxiliary lane code from input card (see card type 4 description for meaning of codes)
LENGTH	I	Hold Area	Link length
LNUM	I	Hold Area	Link number
LTYPE	I	Switch	Link type = (1,2,3) for (surface, ramp, freeway) link
M	I	Hold Area	SIGI approach number
MEANDF	I	Hold Area	Mean desired free-flow speed in- dex in BUFFER array
NUM	I	Hold Area	Relative link number
NUM2	I	Hold Area	Absolute link number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A8/ IPACK, LSTTIM

/A9/ CHEK78

/A2/ ENDNDS, NTOTL

/CONTRL/ IBUF, ICARD, ICDCT, IFN, IFORM,

ILEN, IOP, IRSCT

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/A12/ IGRVAL

/SIZE/ IS, MXLFR, MXLRA, MXLSU, NLFR,
NLRA, NLSU

/A11/ ISTATN, LSTATN, NSTATN

/P1/ NAMES

/A3/ NTOTN, SIGI

3.16 Subroutine LINOUT

Program Narrative

Purpose

LINOUT prints out the link geometry and operation data for all links which was read in by LINKIN.

Assumptions

None.

Approach

First all freeway links are processed and a table is printed for them; then ramp links and finally surface links are likewise processed.

Synopsis

First, all freeway links are processed (if there are no freeway links, then this section is skipped). The table heading is printed. For each freeway link, an output buffer is filled with data unpacked from the LNKF array, and one line per link is printed. Then the early warning sign table is printed, with one line for each link with an early warning sign. Similarly, if there are no ramp links in the network, the ramp section is skipped. Otherwise, the table heading is printed. Then, for each ramp link, an output buffer is filled with data unpacked from the LNKR array, and one line per ramp link is printed. Next, if there are no surface links, the surface section is skipped. Otherwise, the surface table heading is printed. Then, for each link, the output buffer is filled with data unpacked from the LNKS array and one line per surface link is printed.

Finally, if there are any statistical data stations, a statistical data station table is printed, with one line per entry in LSTATN array.

Models and Algorithms

None.

Date, Programmer

July, 1975. Donald Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
BUFFER (47)	I	Hold Area	Unpacked link data, equivalenced to IPACK
I	I	General Do- Loop Index	
ID	I	Hold Area	Downstream node of receiving link
IDWN	I	Hold Area	Downstream node of link
II	I	Hold Area	Grade index
IL	I	Hold Area	Downstream node of link receiving thru traffic
IO	I	Hold Area	Downstream node referred to by early warning sign
IR	I	Hold Area	Temporary storage for unpacking LSTATN data
ISIGN	I	Switch	(0,1) if early warning sign table heading (has not, has) been printed
ITYPE	I	Switch	Link type for stat. data stations = (0,1) for (freeway, ramp) links
ITYPEL	I	Hold Area	Type of last link printed in stat. data station table
IUP	I	Hold Area	Upstream node
J	I	General Do- Loop Index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
J1	I	Hold Area	Absolute link number
L	I	Hold Area	Relative link number
LEN	I	Hold Area	One third of freeway link length
LL	I	Hold Area	Relative link number
LOC	I	Hold Area	Location of statistical data station
ONOFF (2)	I	Hold Area	ONOFF(1) = 2HON, ONOFF(2) = 3HOFF, set in data statement, used in printout tables
OUTBUF (33)	I	Hold Area	Buffer filled with data to be printed out

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/SIZE/ IS, MXLFR, MXLRA, MXLSU, NLFR, NLRA

/All/ ISTATN, LSTATN

/P1/ NAMES

/A3/ NTOTN

3.17 Subroutine LPAK

Program Narrative

Purpose

LPAK reallocates space in the NLV array to allow more space for links of a given type, or simply to compact all the links after all link geometries have been read in.

Assumptions

None.

Approach

LPAK determines the new size and offset parameters needed, and then it readjusts all other parameters.

Synopsis

First, the new maximum size parameters must be determined. If more space is needed for a given link type (if $JJ > 0$), then the new maximum is set to the old maximum plus 10. The maximums for the other types remain the same. If the arrays are to be compacted ($JJ = 0$, all link data has been read in), then the new maximums are set equal to the current maximums. From these maximums, the new offsets for the link types are computed. The remaining space is then allocated to the three vehicle types (70% for freeway vehicles, 10% for ramp vehicles, 20% for surface vehicles); and the vehicle maximums and offsets are computed.

If the surface link array has been expanded, then no further work must be done except to save the new parameters, since surface links come after freeway and ramp links. Next, the pointers must be defined for expanding or compacting so that the blocks of data can be moved to their new locations.

After the NLV data has been shifted, the ENDNDS and NAMES arrays are also shifted to correspond. Finally, the new size and offset parameters are saved.

Models and Algorithms

See the description of the dynamic storage allocation method used in the Final Report, Volume 1.

Date, Programmer

July 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General purpose do- loop index and hold area	
II	I	Do-loop Index	NAMES array index, set in do- loop from 1,6
IND	I	Switch	Set = (1,2) for ramp, surface) operations
J	I	Switch	J = (1,2,3) if expanding (freeway, ramp, surface) link array. J = 0 if compacting all arrays
K	I	General purpose do- loop index and hold area	
L	I	General purpose do- loop index and hold area	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
LC(4)	I	Hold area	If LC(I) > 0, the LC(I) = number coordinates to be shifted. If LC(I) = -1, then area defined by LNU(I) and LOC(I) is zeroed out
LNU(4)	I	Hold area	First word of new location into which data will be shifted, or which will be zeroed out
LO(4)	I	Hold area	First word of old location of data if LC(I) > 0, or last word of area to be zeroed out if LC(I) = 1
LONC(12)	I	Hold area	Equivalenced to LNU, LO, and LC so that all may be initialized at once
M	I	Hold area	Last word of new area or area being zeroed
N	I	Hold area	First word of new area or area being zeroed
NLFRL	I	Hold area	New maximum number of freeway links
NLRA1	I	Hold area	New maximum number of ramp links
NLRO1	I	Hold area	New offset of ramp links
NLSO1	I	Hold area	New offset of surface links
NLSUL	I	Hold area	New maximum number of surface links
NVFO1	I	Hold area	New offset of freeway vehicle
NVFR1	I	Hold area	New maximum number of freeway vehicles

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
NVRAl	I	Hold area	New maximum number of ramp vehicles
NVR01	I	Hold area	New offset of ramp vehicles
NVS01	I	Hold area	New offset of surface vehicles
NVSU1	I	Hold area	New maximum number of surface vehicles

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JJ	I	Switch	Input, JJ=(1,2,3) if expanding (freeway, ramp, surface) links. JJ = 0 if compacting all arrays

Common Blocks and Variables Referenced

```

/ERTRAN/ IEPAR, IER, NPAR
/A1/ MNLV, NLV
/SIZE/ MXLFR, MXLRA, MXLSU, NLFR, NLRA,
      NLRO, NLRO, NLSO, NLSU, NVFO, NVFR, NVRA,
      NVRO, NVSO, NVSU, NWDS
/A2/ NTOTL, ENDNDS
/P1/ NAMES

```

3.18 Subroutine MATCH

Program Narrative

Purpose

MATCH compares all freeway and surface entry links to make sure that no two have the same two lowest-order digits.

Assumptions

None.

Approach

A set of flags is stored, in packed form, indicating whether each of the two-digit numbers from 0 to 99 has appeared in either or both of the 700 and 800 series entry links. When all link geometry data has been read, the flags are checked by the LINKIN subroutine for error conditions.

Synopsis

First, the number (I-1) indicated by the two lowest-order digits is unpacked from the given node number. The word and digit position in the CHEK78 array are computed and the flag is unpacked. The flag is then reset and re-packed into CHEK78 according to its previous status: it is set equal to

- 1 if there exists a node 7(I-1)
- 2 if there exists a node 8(I-1)
- 3 if there exist both nodes 7(I-1) and 8(I-1), an error condition which would be printed out by LINKIN when all link and node geometry data has been processed.

Note: see description of /A9/ for an explanation of the CHEK78 array.

Models and Algorithms

None.

Date, Programmer

July, 1975. Donald Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Hold Area	Lowest-order two digits of node number plus 1
ISS	I	Hold Area	Magnitude in powers of 10 of digit position
J	I	Switch	Set = (1,2,3) if flag was previously set (0,1,2) in CHEK78
N1	I	Hold Area	Word number in CHEK78 for this node number
N2	I	Hold Area	Digit position within word number N1
PICK	I	Hold Area	Flag word unpacked from CHEK78 array, indicating previous status for this node number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
NODE	I	Hold Area	Input, node number between 700 and 899 which must be checked for unique two lower-order digits

Common Blocks and Variables Referenced

/A9/ CHEK78

/SIZE/ IS

3.19 Subroutine PRACT

Program Narrative

Purpose

PRACT serves as an executive routine to read the intersection actuated traffic control input cards.

Assumptions

None.

Approach

This routine reads all input cards for card types 15 and 16 and the first card of card type 17. As soon as it identifies the card type involved, it branches to call routine CTPFF, CTPSX or CTPSV to process the card type 15, 16 or 17, respectively.

Synopsis

The routine identifies the card type and calls the proper subroutine to process it. All input cards for card type 15 and 16 and the first card of card type 17 are read in this routine. An error message is printed when an illegal card type is encountered.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Switch	Code (0,1) if a type 15 or 16 card (has not, has) been read
IB	I	Dummy argu- ment to DRWS	
NCTYP	I	Switch	Code (1,2,3) if card type read was (15, 16, 17)

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, IFORM, IOP

/ERTRAN/ IER, NPAR

3.20 Subroutine PRMSND

Program Narrative

Purpose

PRMSND reads and checks diagnostically the flow rate (type 20) input cards and fills the SNODE array.

Assumptions

PRMSND assumes the card input has been copied to file 13.

Approach

As each card is read, the data is checked and stored, until all type 20 cards have been read.

Synopsis

As each card is read, the card type is checked. If the type is less than 20, an error message is printed and the next card is read. If the type is greater than 20, then processing is finished. If it is type 20, then the data is checked and stored, and error messages are printed as necessary. First, the node numbers are checked for validity, and the corresponding link must be found by FINDL. Then SNODE is primed. For the first subinterval, a new SNODE entry must be created; for subsequent subintervals, the already existing entry must be found. The vehicle type percentages must sum to less than or equal to 100. The flow rate must be non-zero. The lane percentages must sum to less than or equal to 100. If the percentages were not specified, then they must be computed. All SNODE items are computed and stored.

When all type 20 cards have been processed, the last card read is backspaced over (since its type is greater than 20). If it is the first subinterval, each entry link is checked for a corresponding flow rate in the SNODE array.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General purpose do- loop index	
IB	I	Dummy argu- ment to DRWS	
IDN	I	Hold area	Downstream node, from card input
IFR	I	Hold area	Flow rate, from card input
IHO	I	Hold area	Emission headway for previous subinterval
II	I	Pointer	SNODE index for entry link being processed
ILANE	I	Hold area	Number of thru lanes on link
IPSUM	I	Hold area	Sum of vehicle type or lane percentages
IR	I	Hold area	Temporary storage used when unpacking data from input card
IT	I	Switch	Link type: (1,2,3) for (freeway, ramp, surface), 0 if no link found
IUP	I	Hold area	Upstream node, from card input
IY	I	Switch	Link type = (0,1) for (freeway, surface) entry link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
J	I	Do-loop Index	(1) index of data field on card (2) link index
JBUF(13,2)	I	Hold area	Equivalenced to IBUF; data from input card read by DRWS
JJ	I	Hold area	(1) number fields on data card (2) total maximum number links
JP	I	Counter	Position of lane percentage in packed word
JT	I	Switch	Link type of link in SNODE entry = (1,3) for freeway, surface) link
K	I	Do-loop Index	SNODE array index, set in do- loop from 1, NS
L	I	Hold area	Relative link number
LL	I	Hold area	Absolute link number
LX	I	Hold area	Relative link number from SNODE array
NCARD	I	Hold area	Card type
NLFRA	I	Hold area	Total maximum number of freeway and ramp links

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/SIZE/ NLFR, NLRA, NLSU
/CONTRL/ IBUF, ICARD, IFN, IFORM, ILEN, IOP,
NTRN
/ERTRAN/ IEPAR, IER, NPAR
/A2/ ENDNDS
/A3/ NTOTN
/A7/ NS, SNODE

3.21 Subroutine PRSIG

Program Narrative

Purpose

PRSIG reads in and checks diagnostically the type 10 cards and primes the signal and sign control arrays.

Assumptions

PRSIG assumes the card input has been copied to file 13.

Approach

As each type 10 card is read, it is checked diagnostically, and the information is stored in the appropriate arrays.

Synopsis

First, the maximum active node number, MNODE, is computed. Then the next card is read from file 13. If the card type is <10, an error message is printed, and the next card is read. If it is >10, then the final checking will be performed (to be described later). If it is a type 10 card, the node number is checked for validity. If the format is for sign and fixed time signals, the appropriate checks are made. The approach links must be counted -- there must be at least one. The control codes must be valid, and a code must be specified for every approach. For each signal code, a non-zero phase duration must be specified. The signal codes are then stored in the SIG array; the phase durations in the NPHS array.

Then, for signals, the reference offset must be checked so that the initial interval and time remaining can be stored in the SIGT array.

Next, for both signs and signals, the node and link consistency is checked for all approaches. As each link is found, the current signal is stored in the appropriate link

array. Next, it is checked that all turning movements are serviced by at least one interval for signals, or that they have a valid sign code. SIGI, containing the approaches to each node, is reordered to be consistent with the card 10 input, and all approaches in SIGI are checked for valid signal. Then the next type 10 card is read and processed in the same way.

If the format is for actuated controls, then the SIG array is filled appropriately, and the actuated control parameters are stored in the IACPAR array.

When all type 10 cards have been read, we check whether at least one was found; if not, a message is printed. Finally, consistency between the SIG (signal) and SIGI (approach) arrays are checked (each node must have entries in both or neither arrays). The phase duration for signed intersections is set to the maximum cycle length for signals, or 80 seconds if no signals. The last card read is backspaced over, and control returns to the calling routine.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	General purpose do- loop index and hold area	
IB	I	Dummy argu- ment to DRWS	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
ICODE	I	Hold area	Current (initial) signal code
ICYCTM	I	Hold area	Cycle time
II	I	Do-loop Index	Approach number, set in do-loop from 1,4
IL	I	Hold area	Absolute link number
INODE	I	Hold area	Node number, from input card type 10
IREF	I	Hold area	Reference offset, from input card type 10
ISAV	I	Hold area	Index of current (initial) signal code
IT	I	Switch	(1,2,3) for (freeway, ramp, surface) link, 0 if no link
ITL	I	Hold area	Sum of phase durations up to interval of reference offset
ITP	I	Switch	= (0,1,2) for (freeway, ramp, surface) vehicles
ITURN	I	Switch	Turn code = (0,1,2) for (left, thru, right)
J	I	General purpose do- loop index and hold area	
JBUF(5,6)	I	Hold area	JBUF(I,J), I = 1,4 contains signal for approach I, interval J. JBUF(5,J) contains dura- tion for interval J

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JCODE	I	Switch	(0,1) if controls (do not, do) exist at node
JNODE	I	Hold area	Upstream node of approach link
K	I	Do-loop Index	Approach number, set in do-loop from 1,4
L	I	Hold area	Relative link number
LL	I	Hold area	Absolute link number
LP	I	Hold area	Turn code
LS	I	Hold area	Relative link number from SIGI array
NAPPR	I	Hold area	Number of approaches
NCARD	I	Counter	Number of type 10 cards read in
NIVL	I	Counter	Number of intervals in signal cycle
NUM	I	Hold area	Temporary storage of packed signal codes
N1	I	Hold area	Number of approaches plus 1

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

```

/A2/ ENDNDS
/A6/ IACPAR, IANOD, MANOD, MNODE, NPHS,
    SIG, SIGT
/CONTRL/ IBUF, ICARD, IFN, IFORM, ILEN, IOP
/ERTRAN/ IEPAR, IER, NPAR
/SIZE/ IS
/S2/ MCYCL
/A3/ NTOTN, SIGI

```

3.22 Subroutine SIGOUT

Program Narrative

Purpose

SIGOUT prints out the traffic signal data input by PRSIG.

Assumptions

None.

Approach

As SIGOUT loops through each node, it prints out the appropriate information.

Synopsis

First, the page heading is printed. Then SIGOUT loops through all nodes in the network, skipping those that are not defined. It prints different formats for fixed time signal and sign control and for actuated signals. For fixed time signals and signs, the number of approaches is computed. Then the approach links are formed as well as the upstream and downstream nodes. Then the subheading is printed. The percentage duration, offset, and percentage offset for each interval is computed. Finally, all this information is printed out along with node number, interval number, phase duration, and signal code facing each approach. An additional message is printed indicating those nodes under sign control.

For actuated signals, the node number, actuated node number (index to IACPAR array), algorithm number, and parameter list are printed, along with a message indicating the node is under traffic-actuated control.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Node number, set in do-loop from 1, MNODE
IAL	I	Hold Area	Actuated algorithm number
IAN	I	Hold Area	Actuated node number
ICOD (6,4)	I	Hold Area	ICOD (I,J) contains unpacked sig- nal code for interval I, approach J
ICYCTM	I	Hold Area	Cycle length
IC2	I	Hold Area	One-half of cycle length
IDP (6)	I	Hold Area	Percentage of cycle length of each interval
IFP (6)	I	Hold Area	Percentage offset of each interval
II	I	Do-loop Index	Approach number, set in do-loop from 1,4
ILNK	I	Hold Area	Link number of approach link
INOD(4)	I	Hold Area	Upstream nodes of approach
IOFF(6)	I	Hold Area	Offset of each interval
IOFFR	I	Hold Area	Offset of interval

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IR	I	Hold Area	Temporary storage used to unpack signal codes
IREF	I	Hold Area	Reference offset of cycle
ITL	I	Hold Area	Current active interval
ITYPE	I	Hold Area	Link type of approach link
J	I	Do-loop Index	Approach number, set in do-loop from 1, NAPPR
K	I	Do-loop Index	Interval number, set in do-loop from 1, NIVL
KK	I	Do-loop Index	Interval number, set in do-loop from NN, NIVL
NAPPR	I	Hold Area	Number of approaches
NIVL	I	Hold Area	Number of intervals
NN	I	Hold Area	Interval after current active interval

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/A6/ IACPAR, MNODE, NPHS, SIG, SIGT

/SIZE/ IS, NLFR, NLRA

/A3/ SIGI

3.23 Subroutine SUROUT

Program Narrative

Purpose

This routine prints a table of detector data from the DTCTR array.

Assumptions

None.

Approach

Detector data is unpacked from common storage and formatted for output.

Synopsis

Output table headings are written and the routine begins a loop over all freeway links. If a link has detectors on it which have station numbers, the detector data is output. After the last detector having a station number on a link is located, the loop over freeway links continues.

The loop is entered again to search for all detectors which have not been assigned station numbers. After this second pass, a glossary of detector types is output.

Models and Algorithms

None.

Date, Programmer

November, 1976. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Used to loop over a sequence of freeway links
IDTC	I	Hold Area	Detector number
ILE	I	Hold Area	Detector loop length
ILN	I	Hold Area	Lane number
IR	I	Hold Area	Temporary storage for packed detector word
ITYPE	I	Hold Area	Detector type
KD	I	Hold Area	Detector counter
L	I	Hold Area	Freeway link number
LINDEX	I	Hold Area	Index to freeway link data
NFR	I	Hold Area	Number of freeway links
NNFR	I	Hold Area	Index to sequence of freeway links

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/NLV
/A2/ENDNDS,NTOTL
/A4/DTCTR,IDET
/RANK/IFRANK
/SIZE/NLFO

3.24 Subroutine SURVIN

Program Narrative

Purpose

SURVIN reads in the surveillance detector cards (type 25), checks them diagnostically, and stores the data in the detector array.

Assumptions

SURVIN assumes the card input has been copied to file 13.

Approach

As each card is read, it is checked diagnostically, and the data is stored in the detector array.

Synopsis

As each card is read, the card type is checked. If the card type is greater than 25, then the last card is backspaced over, and processing is finished. If it is less than 25, an error message is printed and processing goes on to the next card. If it is type 25, the data is checked. The upstream and downstream nodes must be valid and must not define a surface entry link. The link must be a valid link defined by the link geometry cards.

Next, all the non-empty detector fields are checked. The data is unpacked. A valid location in the DTCTR array must be found. If this is the first detector for this link, the location must be stored in the link array. If not, it must be linked to the previous last detector. The detector type must be valid. The loop length must be non-negative. The lane and longitudinal placement must be valid, taking into consideration whether there are auxiliary lanes, turn pockets, etc.

When all checks have been made, the data is stored in the DTCTR array. If the detector type is downstream loop of coupled pair, then a DTCTR entry must be generated for the

upstream loop. Otherwise processing goes on to the next card.

Models and Algorithms

None.

Date, Programmer

July 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	(1) Hold area (2) Do-loop index	(1) upstream node from input card (2) input card field index
IA(4,6)	I	Hold area	Equivalenced to IBUF; detector data from input card
ICD	I	Hold area	(1) pointer to auxiliary free-way lane data (2) number of turn pockets on surface link
IDTPE	I	Hold area	Detector type from input card
IFE	I	Hold area	Index of first detector for link
II	I	Hold area	Upstream node from input card
ILE	I	Hold area	Index of last detector for this link
ILN	I	Hold area	Pointer to next detector on link, 0 if no more detectors
IPARM(47)	I	Hold area	Unpacked link data

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
ISEPR	I	Hold area	Distance between loops of coupled pair, from input card
ITP	I	Switch	Link type = 1,2,3) for (free-way, ramp, surface) links
J	I	Hold area	Downstream node from input card
JCD	I	Switch	(0,1) if lane specified (is, is not) valid
L	I	Hold area	Relative link index, computed by FINDL
LAT	I	Hold area	Lane specified on input card
LL	I	Hold area	Absolute link index
LLEN	I	Hold area	Effective loop length, from input card
LONG	I	Hold area	Longitudinal placement of detector, from input card

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/ DTCTR, IDET, NDET
 /CONTRL/ IBUF, ICARD, IFN, IFORM, IOP
 /ERTRAN/ IEPAR, IER, NPAR
 /A3/ NTOTN

3.25 Subroutine TABCON

Program Narrative

Purpose

TABCON prints a table of contents for a case data tape.

Assumptions

TABCON assumes the user has designated previously created case data tape as unit 8.

Approach

TABCON loops through all cases until the type 62 terminator is found, signaling the end of data.

Synopsis

First, the case data tape is rewound. Then the table heading is printed out. For each case on tape, the first two card images are printed in the table. Then all cards are skipped until a type 61 or type 62 terminator is found. Type 61 means that another case follows; if so, it is processed like the first one. Type 62 means no more data is on the tape, so it is rewound, and control returns to the calling program.

Models and Algorithms

None.

Date, Programmer

July, 1975. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Card numbers, set in do-loop index from 1,2
IC	I	Counter	Case number
ID	I	Hold area	Card type
J	I	Do-loop Index	Card image work, set in read and write statements from 1,40

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, INLOC, IOP
/A1/ NLV

3.26 Subroutine TURNIN

Program Narrative

Purpose

TURNIN reads type 8 input cards (link turning movement cards), checks them diagnostically, and stores the data in the appropriate arrays.

Assumptions

TURNIN assumes the input cards have been copied to file 13.

Approach

As each card is read, it is checked diagnostically and the data is stored, until all type 8 cards have been read.

Synopsis

DRWS is called to read the next card. If an end-of-file is encountered or if the card type is less than 8, a message is printed and control returns to the calling program. If the card type is greater than 8, then that record is backspaced over and control returns to the calling program. If it is type 8, then each of five possible fields is processed as follows: the downstream node must be valid and not an exit node. The turning percentages must be non-negative. The sum of the three percentages must be greater than zero. If the turning movements are expressed in absolute numbers, they are converted to percentages. The upstream node must be valid. When all these tests have been passed, FINDL is called to determine the link number, and UNPAK is then called to unpack the appropriate link data. A branch is taken depending on link type, as follows:

Freeway links: if there is an exit ramp from this link, then a non-zero percentage must be specified for right or left turn for a right or left exit ramp respectively. If there is no exit ramp, then traffic must be 100% through.

Ramp links: if ramp has two downstream movements, then non-zero percentage must be specified for each. If there is only one movement, then 100% must be specified for that movement.

Surface links: for each downstream receiving link, a non-zero turn percentage must be specified, and for each percentage specified there must be a receiving link.

Finally, the turning percentages are stored and the data is repacked into the link array.

Models and Algorithms

None.

Date, Programmer

July, 1975. Donald Goldman

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
BUFFER (47)	I	Hold Area	Buffer containing unpacked link data, equivalenced to IPACK
I	I	General Purpose Do-Loop Index	
IB	I	Dummy Argument to DRWS	
IBUF8 (5,5)	I	Hold Area	Input card data by field, equivalenced to IBUF
IDWN	I	Hold Area	Downstream node
IT	I	Switch	(0,1,2,3) for (no link, freeway link, ramp link, surface link)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IUP	I	Hold Area	Upstream node
I1	I	Hold Area	Index to receiving link
I2	I	Counter	Number of non-zero turning percentages specified
I3	I	Counter	Number of receiving links specified
K	I	Do-Loop Index	Field on input card, set in do-loop from 1 to 5
LNUM	I	Hold Area	Relative link number
SUM	I	Hold Area	Sum of turning percentages

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, ICARD, IFN, IFORM, ILEN, IOD

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/A8/ IPACK

/SIZE/ IS

/A3/ NTOTN

4. LIS OVERLAY

The purpose of the LIS overlay is to load simulation case data into data arrays and scalars in preparation for execution of a simulation subinterval. LIS may also retrieve and execute error free data sets stored on the Case Data Tape.

Because the PORGIS and LIS Overlays perform largely identical functions, all subroutines in LIS have an analogous routine in PORGIS. Therefore, the documentation of those routines in PORGIS also applies to their analogs in LIS. The following is a list of those subroutines in LIS which have corresponding routines in PORGIS:

<u>LIS Overlay</u>	<u>PORGIS Overlay</u>
LIS	PORGIS
LCLRAL	CLRALL
LCTPFF	CTPFF
LCTPSV	CTPSV
LCTPSX	CTPSX
LDETGE	DETGEN
LFLOOU	FLOOUT
LINACT	INACT
LINCIN	INCIN
LINCOU	INCOUT
LINT1	INT1
LIMBDO	IMBEDO
LIMBED	IMBED
LLINKI	LINKIN
LLINOU	LINOUT
LLPAK	LPAK
LPRACT	PRACT
LPRMSN	PRMSND
LPRSIG	PRSIG
LSIGOU	SIGOUT
LSUROU	SUROUT
LSURVI	SURVIN
LTURNI	TURNIN

5. SIFT OVERLAY

This overlay performs all of the required simulation activities. This overlay contains two sub-overlays; HICON, which performs those activities required every time-step; and LOCON, which performs those activities required only periodically.

5.1 Program SIFT

Program Narrative

Purpose

SIFT is the main supervisor for the actual simulation of traffic in each subinterval. It calls the high- and low-activity overlays at the appropriate times to perform all necessary initialization, simulation, and cleanup operations as well as certain tape and file operations for the other program modules requiring data from the simulation.

Assumptions

SIFT assumes the PORGIS and LIS overlays have been called to provide it with a complete, error-free set of input data defining the network geometry as well as subinterval-specific control parameters.

Approach

SIFT loops over all time steps for one subinterval. At each time step, various counters are examined and decisions made as to whether it is time to call each lower level overlay.

Synopsis

First, the LOCON overlay is called to perform subinterval-specific initialization. Then, if there are no actual freeway vehicles in the system, a dummy freeway vehicle must be defined to satisfy the requirements of the freeway model. The same is done for ramp vehicles.

Then the main simulation control loop is entered; each time-step is simulated within this loop. HICON is called to simulate traffic for one time-step. Then a branch is taken depending on whether it is fill-time. If it is fill-time, and fill-time is to continue even if equilibrium is reached, then simulation continues with the next time-step. If not, then if the fill-time evaluation period is over, LOCON is called to check whether equilibrium has been reached; if so, then the first subinterval will be executed; if not, fill-time continues. If the evaluation period is not over, then, fill-time continues. If it is not fill-time but rather, an actual

subinterval, then, if the contour plot output interval is over, LOCON is called to write out the contour plot data to tape. If the standard output interval (if any) is over, or if the subinterval is over, then LOCON is called to print the standard output tables. If intermediate output was requested, and the intermediate output interval is over, then LOCON is called to print the intermediate output tables. This ends the main control loop.

The previous subinterval duration is saved for future use. Then, if the control loop has just performed fill-time and simulation was not inhibited by some error condition, then the first subinterval is executed; if it was inhibited, then the case is aborted. If the subinterval simulation was inhibited by some error condition, then the case is also aborted. If the subinterval was successful and it is the last subinterval, then the last record flag and delimiter record must be written to the plot tape, if any. If there is to be a FUEL overlay execution, the end of subinterval and/or end of run flags must be written to the FUEL temporary file as well as to the FUEL tape if one is being generated.

Finally, the NEXCAL flag must be set to indicate the next overlay depending on whether the simulation is over and whether the user has requested any statistical modules.

Models and Algorithms

None.

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Time-step index, set from 1 to NIDT (number of time-steps in fill-time or subinterval)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IB	I	Dummy Argument to DRWS	
IHAF	I	Hold Area	Dummy values packed into dummy freeway and ramp vehicles
J	I	Hold Area	Contour plot interval duration; simulation time, seconds.
JBUF (101)	I	Hold Area	Temporary storage of last plot record so that last record flag may be written, equivalenced to NF array.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, ICPOP, ICTYPE, IDMN, IDSEC,
IFILGO, IFN, IFORM, IGO, ILEN, ILSUB,
IOP, IRSLT, ISECI, ISTAT, ITPOP,
NPRNT

/S2/ ICOINT, ICYCPO, IDT, IDTS, IMCYCL,
IPSL, ITIME, NIDT

/ERTRAN/ IERCT

/VPOINT/ IFPT, IRPT, ISPT

/IPLTN/ IPLTN

/S3/ ISRET, ISSEQ

/VBUF/ NF

/CFUEL/ JFUEL, NU23

5.2 Subroutine FINDFV

Program Narrative

Purpose

FINDFV locates an empty slot in the freeway vehicle array.

Assumptions

None.

Approach

The freeway vehicle array is searched, and space is reallocated if necessary, until an empty space is found.

Synopsis

The freeway vehicle array is searched, starting at the point where the last empty slot was found, until an empty slot is found or the end of the array is reached. If no slot is found, a second search is made from the beginning of the array. If no slot is found, then VPAK is called to allocate more space to the freeway vehicle array. If no more space can be allocated, then an error message is printed and the simulation is aborted. Otherwise a third search is made to find the new empty slot provided by VPAK.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	First vehicle slot to be examined
IFLAG	I	Switch	(1,2) for (first, second) pass over vehicle array
K	I	Do-Loop Index	Vehicle array index
MAX	I	Hold Area	Last vehicle slot to be examined
N	I	Hold Area	Maximum number of vehicles allowed before reallocation by VPAK

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NEWV	I	Hold Area	Output, empty vehicle slot number

Common Blocks and Variables Referenced

/ERTRAN/ IER, NPAR

/VPOINT/ IFPT

/SIZE/ MXVFR, NVFO, NVFR

/A1/ VF

5.3 Subroutine FINDRV

Program Narrative

Purpose

FINDRV locates an empty slot in the ramp vehicle array.

Assumptions

None.

Approach

The ramp vehicle array is searched, and space is reallocated if necessary, until an empty space is found.

Synopsis

The ramp vehicle array is searched, starting at the point where the last empty slot was found, until an empty slot is found or the end of the array is reached. If no slot is found, a second search is made from the beginning of the array. If no slot is found, then VPAK is called to allocate more space to the ramp vehicle array. If no more space can be allocated, then an error message is printed and the simulation is aborted. Otherwise a third search is made to find the new empty slot provided by VPAK.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	First vehicle slot to be examined
IFLAG	I	Switch	(1,2) for (first, second) pass over vehicle array
K	I	Do-Loop Index	Vehicle array index
MAX	I	Hold Area	Last vehicle slot to be examined
N	I	Hold Area	Maximum number of vehicles allowed before reallocation by VPAK

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NEWV	I	Hold Area	Output, empty vehicle slot number

Common Blocks and Variables Referenced

/ERTRAN/ IER, NPAR

/VPOINT/ IRPT

/SIZE/ MXVRA, NVRA, NVRO

/A1/ VR

5.4 Subroutine FINDSV

Program Narrative

Purpose

FINDSV locates an empty slot in the surface vehicle array.

Assumptions

None.

Approach

~~The surface vehicle array is searched, and~~
space is reallocated if necessary, until an empty space is found.

Synopsis

The surface vehicle array is searched, starting at the point where the last empty slot was found, until an empty slot is found or the end of the array is reached. If no slot is found, a second search is made from the beginning of the array. If no slot is found, then VPAK is called to allocate more space to the surface vehicle array. If no more space can be allocated, then an error message is printed and the simulation is aborted. Otherwise a third search is made to find the new empty slot provided by VPAK.

Models and Algorithms

None.

Date, Programmer

January 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	First vehicle slot to be examined
IFLAG	I	Switch	(1,2) for (first,second) pass over vehicle array
K	I	Do-Loop Index	Vehicle array index
MAX	I	Hold Area	Last vehicle slot to be examined
N	I	Hold Area	Maximum number of vehicles allowed before reallocation by VPAK

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NEWV	I	Hold Area	Output, empty vehicle slot number

Common Blocks and Variables Referenced

/ERTRAN/ IER, NPAR

/VPOINT/ ISPT

/SIZE/ MXVSU, NVSU, NVSO

/A1/ VS

5.5 Subroutine FRSTV

Program Narrative

Purpose

FRSTV determines the vehicle farthest downstream in a given lane of a given link.

Assumptions

FRSTV assumes that the input parameters (farthest upstream vehicle, link, lane, link type) are all valid parameters.

Approach

Successive leaders of the given vehicle are examined until the one with no leader in the given link is found.

Synopsis

If FRSTV was called from the FMOVE routine, a flag must be set and the upstream vehicle number must be made positive. A branch is taken depending on link type, so that the appropriate arrays may be searched:

For freeway links: the right leader, MA, of the given vehicle M, is found. The current or right-hand lane is found. If the given vehicle is in this lane, then the leader has been found; otherwise the left leader must be examined (recall that a freeway vehicle can be associated with two lanes while it is changing lanes). If the leader vehicle number is 0 (i.e., no leader), then the given vehicle is the farthest downstream vehicle. If the link number of the leader is different from the given link, the given vehicle is likewise the farthest downstream vehicle. If not, the leader becomes the given vehicle and the process is repeated until the farthest downstream vehicle is found. When found, certain parameters must be unpacked and stored in the /FRSTVP/ common block (position, speed, acceleration, vehicle length) for use by certain calling routines.

For ramp links: the leader of the given vehicle is examined; if it is 0, or if the leader's link is not the same as the given link, or if the leader lane number is greater than 2 (meaning it is a ramp vehicle about to discharge to a freeway), the given vehicle is the farthest downstream vehicle. Otherwise, the leader becomes the given vehicle and the process is repeated until the farthest downstream vehicle is found.

Similarly, for surface links: each successive leader is examined until one is found with no leader, or whose leader is in a different link.

When the vehicle is found, the farthest downstream vehicle parameter is set. If the call was from FMOVE, the leader in the next link of the farthest downstream vehicle is also returned to the calling routine.

Note to user: in the present version, the variable IPOST is always equal to -1.

Program Variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
FMFLAG	I	Switch	(0,1) if call (was not, was) from FMOVE
IHAF	I	Hold area	Temporary storage of link and vehicle parameters
IK	I	Hold area	Link number
IKTYP	I	Hold area	Link type
IN	I	Hold area	Lane number
L	I	Hold area	Leader vehicle number
LK	I	Hold area	Link number of leader vehicle

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LN	I	Hold area	Lane number or right hand lane if vehicle is changing lanes
M	I	Hold area	Candidate farthest downstream vehicle
MA	I	Hold area	Leader vehicle of candidate vehicle

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIK	I	Hold area	Input, link number
IIKTYP	I	Switch	Input, (0,1,2) for (freeway, ramp, surface) link
IIN	I	Hold area	Input, lane number
MM	I	Hold area	Output, farthest downstream vehicle
MMA	I	Hold area	Input, upstream vehicle; output, leader of downstream vehicle, if any, if called from FMOVE

Common Blocks and Variables Referenced

/A8/CLEN

/FRSTVP/IJFAC, IJFLEN, IJFPOS, IJFSP, IPOST

5.6 Subroutine LASTLK

Program Narrative

Purpose

LASTLK determines the link number type and lane upstream of a given link and lane.

Assumptions

LASTLK assumes that the given link is a freeway link.

Approach

LASTLK finds the link whose receiving link is the given link and then calls FINDLN to determine the lane whose receiving lane is the given lane.

Synopsis

First, LASTLK checks all freeway links to see if any has the given link as its receiving link. If none is found, it then checks all ramp links for the same purpose. If no freeway or ramp link can be found upstream of the given link, all output parameters are set to zero, and control returns to the calling routine.

If a freeway or ramp link has been found, FINDLN is called to determine the lane alignment scheme of the two links. The upstream link and link type parameters are also set by FINDLN. The downstream lanes in LARRAY (output from FINDLN) are searched for a match with the given lane. When it is found, the corresponding upstream lane is examined: if it is positive, it is a freeway link and processing is finished. If negative, the remaining downstream lanes must be checked to see if the given lane is also fed by a freeway link. If it is fed by both ramp and freeway links, the freeway link is returned as the result. If it is fed by only a ramp, the ramp is returned as a result; the ramp link array must be searched to determine the ramp link number.

If the downstream lane cannot be found in the LARRAY, all parameters are set to zero to indicate an error condition.

Control then returns to the calling program.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Downstream lane index; also, ramp link index
II	I	Hold area	Index of next LARRAY location after first match found
IK	I	Hold area	Downstream link number
IL	I	Hold area	Downstream lane number
K	I	Do-loop index	Freeway or ramp link index; also, LARRAY lane index
LARRAY (2,11)	I	Hold Area	Lane alignment scheme, output from FINDLN, LARRAY (2,K) is receiving lane of LARRAY (1,K) if any
LTEMP	I	Hold area	Thru receiving lane

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIK	I	Hold area	Input, downstream link number
IIL	I	Hold area	Input, downstream lane number
LASL	I	Hold area	Output, upstream link number feeding IIK
LKTYP	I	Switch	Output, (0,1) for (freeway or no, ramp) link
LPREV	I	Hold area	Output, upstream lane number feeding IIL

Common blocks and variables referenced

/SIZE/MXLFR, MXLRA, NLFR, NLRA

5.7 Subroutine LASTV

Program Narrative

Purpose

LASTV determines the farthest upstream vehicle in a given lane of a given link as well as identifying certain characteristics of the lane.

Assumptions

LASTV assumes the input parameters (link, link type, lane number) are all valid.

Approach

The farthest upstream vehicle is determined from the link array according to link type.

Synopsis

First, if no link or lane was specified, all output parameters are set to zero and processing is over. Otherwise, a branch is taken according to link type:

For freeway links: for thru lanes, the farthest upstream vehicle is found directly in the link array. For auxiliary lanes, it must be determined whether the lane is the first or second auxiliary lane before the information can be found. The KAUX and LTYP flags are set according to the lane attributes.

For ramp and surface links, the farthest upstream vehicle is found directly in the ramp or surface link arrays. The LTYP and KAUX flags remain equal to zero.

The farthest upstream vehicle and lane type are returned to the calling routine.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews

Program variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IK	I	Hold area	Link number
IKTYP	I	Switch	(0,1,2) for (freeway, ramp, surface) links
IL	I	Hold area	Lane number
ILA	I	Hold area	Pointer to farthest upstream vehicle
L3	I	Hold area	First auxiliary lane code
MLT	I	Hold area	Farthest upstream vehicle
LTYP	I	Switch	(0,1,2,3) for (acceleration lane or ramp or surface link, deceleration lane, regular freeway lane)

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIK	I	Hold area	Input, link number
IIKTYP	I	Switch	Input, link type (0,1,2) for (freeway, ramp, surface) link
IIL	I	Hold area	Input, lane number
J	I	Hold area	Output, farthest upstream vehicle

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KAUX	I	Switch	Output, (0,1,2) for (no auxiliary lane, auxiliary lane 1, auxiliary lane 2)
M	I	Switch	Output, lane type, (0,1,2,3) for acceleration lane or ramp link or surface link, deceleration lane, acceleration-deceleration lane, regular freeway lane)

Common blocks and variables referenced

None

5.8 Subroutine RANDOM

Program Narrative

Purpose

RANDOM generates a pseudo-random number based upon a stored "seed" value.

Assumptions

None.

Approach

The random numbers are produced by a multiplicative congruential pseudo-random number generator.

Synopsis

During each call to RANDOM, the random number seed is revised and a two-digit integer, three-digit integer and real (between zero and one) random numbers are produced.

Models and Algorithms

A multiplicative congruential pseudo-random number generator is employed.

Date, Programmer

June 1975. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ANR	R	Hold Area	Contains the value of the two-digit random number (in floating point)
I	I	Hold Area	Stores temporary value used in revising the random number seed

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NR	I	Hold Area	Output, two-digit random number
MR	I	Hold Area	Output, three-digit random number
RNU	R	Hold Area	Output, a random number between zero and one

Common Blocks and Variables Referenced

/CONTRL/ RNSEED

5.9 Subroutine VPAK

Program Narrative

Purpose

VPAK reallocates the space available to the three vehicle arrays so that each type is allowed space proportional to its current percentage of total vehicles, so that if there are very few vehicles of a given type, there will not be unnecessary space allocated to that type.

Assumptions

None.

Approach

New vehicle maximums and offsets are computed according to the proportion of total available space required by each type. Then the data currently stored is shifted into its new position.

Synopsis

First, the total number of words available and the current number being used by each vehicle type are computed. If the current numbers are less than 2, they are set to a minimum of 2. Then new maximums for each vehicle type are computed by assigning available space in proportions to the amount of space currently being used by each vehicle type. If any of the new maximums are less than the current number of vehicles, an error message is printed and the job is aborted. Otherwise, the KPOS array is filled with pointer information allowing the ramp and surface vehicle arrays to be repositioned correctly. Then each set of data is shifted forward or backward according to the pointers in the KPOS array. Finally, any unused space is zeroed out, the new maximum size parameters are stored, and control returns to the calling routine.

Models and Algorithms

See the INTRAS Final Report, Volume I, for a complete description of the dynamic storage allocation scheme.

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Index to KPOS, first or second data area to be shifted
ITJM	I	Hold Area	Simulation time, seconds
J	I	Switch	(1,-1) if data is to be shifted (backward, forward)
K	I	Hold Area	Pointer to location into which data is to be shifted
KLFU	I	Hold Area	Current number of freeway vehicle words being used
KLRU	I	Hold Area	Current number of ramp vehicle words being used
KLSU	I	Hold Area	Current number of surface vehicle words being used
KLU	I	Hold Area	Current total number of vehicle words being used
KPOS (3,2)	I	Hold Area	KPOS(1,I) = pointer to first word to be shifted KPOS(2,I) = pointer to location into which first word will be shifted KPOS(3,I) = pointer to last word to be shifted I = (1,2) for (ramp, surface) or (surface, ramp) depending on which must be shifted first to avoid destroying data

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KSA	I	Hold Area	Total number of words available for all vehicle types
L	I	Hold Area	Pointer to word being shifted
LL	I	Hold Area	Pointer to last word to be shifted
NVFRN	I	Hold Area	New maximum number of freeway vehicles
NVRAN	I	Hold Area	New maximum number of ramp vehicles

NVRON	I	Hold Area	New ramp vehicle offset
NVSON	I	Hold Area	New surface vehicle offset
NVSUN	I	Hold Area	New maximum number of surface vehicles

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/ERTRAN/ IEPAR, IER, NPAR

/S2/ ITIME

/A1/ MNLV, NLV

/SIZE/ MXVFR, MXVRA, MXVSU, NVFO, NVFR, NVRA,
NVRO, NVSO, NVSU, NWDS

5.10 Program HICON

Program Narrative

Purpose

HICON is the supervisor program for the high-activity simulation routines: it calls the routines to simulate traffic during one time-step, from generating new vehicles to performing final bookkeeping activities.

Assumptions

None.

Approach

HICON calls a different routine to perform each function depending on whether it is fill-time and whether the freeway time-step is over.

Synopsis

First, if a trajectory plot tape is to be written, the first record at time = 0 must be generated by TPTOUT. MOEV is called to process non-freeway vehicles for one time-step, and SVEH is then called to generate new non-freeway vehicles.

If there are freeway vehicles in the network and the freeway time-step is over, then FMAIN is called to process freeway vehicles, the freeway simulation time is incremented, and, if it is not fill-time, INCDAT is called to output data to the incident data tape.

UPSIG is called to update the signalization for the next time-step. The simulation time is incremented. CLNUP is called to perform necessary bookkeeping activities.

If it is not fill-time, and a trajectory plot tape is to be written, and the trajectory plot output interval is over, then TPTOUT is called to generate trajectory plot records for this time-step.

If it is not fill-time, and it is the last subinterval and the last surface time-step, then INCDAT is called to

write an end of case flag on the incident data tape.

Control finally returns to SIFT, the calling program.

Models and Algorithms

None.

Date, Programmer

January, 1976. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Hold Area	Trajectory plot output interval (10ths of a second)

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/S2/ IDT, IDTF, IDTS, IFDT, ITIME, ITIMEF,
ITRINT, NIDT

/FORINC/ IINC

/S3/ ISRET

/CONTRL/ ITPOP, LASUBF, MXINIT

/SIZE/ MXLFR

5.11 Subroutine ACTFZ

Program Narrative

Purpose

To activate those phases as indicated in the argument list and to calculate the duration of the added initial interval, if any.

Assumptions

Assumes DETECT routine has provided pertinent information about the detector.

Approach

This routine activates phases that were specified in the calling arguments list by setting the status code of these phases to reflect the initial interval. If an overlapping phase is activated, the routine sets its status code to reflect the green service interval directly. For those phases that have just been activated, the routine also calculates the duration of the added initial interval, if appropriate.

Synopsis

The routine activates one phase at a time if more than one phase is to be activated. When the phase to be activated has an overlapping phase, the overlapping phase is also activated. Routine ASIG is called to set link-specific signal codes to reflect the phase and its overlapping phase, if any, which has just been activated. The routine then proceeds to calculate the necessary duration of the added initial interval, if any, to account for those actuations which took place while this particular phase was still inactive. For a non-actuated phase and for an actuated phase with a fixed initial interval, the above calculation is by-passed.

Finally, the status code for the phase in question is set to reflect the entering of an initial interval. For an overlapping phase, the status code is set to reflect the service green interval directly.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Detector index, set in loop from 1 to 8 in intervals of 2
IAD	I	Hold Area	Initial actuation data
IIC	I	Hold Area	Initial interval code
IID	I	Hold Area	Initial interval data ring
IR	I	Hold Area	Ring
J	I	Hold Area	Maximum number of calls for phase
JJ	I	Hold Area	Status code for phase NP
M	I	Hold Area	Activated comparison phase
MAXCLS	I	Hold Area	Maximum number of calls for phase NP
MININ	I	Hold Area	Minimum initial interval

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, activated phase on Ring 1
I3	I	Hold Area	Input, activated phase on Ring 2

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT6/ NACNT

/ACT8/ NACDS

/ACT12/ NASTAT

5.12 Subroutine ADVANC

Program Narrative

Purpose

ADVANC is called to determine if a specified vehicle has passed an advance warning sign during its current movement and, if so, to assign a downstream turning movement.

Assumptions

It is assumed that the elements of the FORINC COMMON block have been primed with information defining the current vehicle movement. It is also assumed that the NF and IPACK arrays have been primed with the subject vehicle and link parameters.

Approach

ADVANC first determines if there is an advanced warning sign on the link and then whether it has been passed. If so, the turning percentage for the downstream ramp is applied to determine this vehicle's turning movement.

Synopsis

First, the vehicle is examined to determine if it has either just left the freeway or is already assigned to exit. In either case, no further action is taken.

If an advanced warning sign exists on this link, then it is determined whether the current vehicle movement passed this sign. If not, ADVANC returns to the calling program.

The turn percentage, at the referenced off-ramp, is extracted from the freeway link array for the link approaching the off-ramp. RANDOM is called to obtain a random number to be used in determining the new turn code. An effective turn percentage is calculated based upon the input turn percent and the history of movements thus far. The random number is compared to this effective turn

percentage and, if it is less, then the vehicle is flagged as destined for the ramp in question.

For each such ramp-bound vehicle, its desired lane is revised so as to correspond to the extreme through lane on the ramp side of the freeway.

Models and Algorithms

None.

Date, Programmer

June 1976. David Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General purpose Do-loop index and hold area	
IHAF	I	General purpose Hold area	
INTENT	I	Hold area	Node number downstream at which vehicle will exit freeway
IPO1	I	Hold area	Vehicle position (at beginning of movement) relative to upstream end of link being processed, feet
IPO2	I	Hold area	Vehicle position (at end of movement) relative to upstream end of link being processed, feet
J	I	Hold area	Number of approach to node lo- cating off-ramp which is the freeway approach
JHAF	I	Hold area	Link length, feet
KDEFA	I	Hold area	Actual percent of vehicles that has turned
KQ	I	Hold area	Random number
KTOT	I	Hold area	Total number of vehicles enter- ing freeway link (during subin- terval) whose downstream node is referenced by early warning sign

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KTURN	I	Hold area	Number of vehicles that have turned so far
L	I	Hold area	Freeway link number
LDOWN	I	Hold area	Freeway link number whose downstream node is referenced by early warning sign
LEFT	I	Switch	Code (0,1) if off-ramp is (right hand, left hand)
LINDEX	I	Hold area	LNKF array index for subject freeway link
LSIN	I	Hold area	Location of early warning sign on link, feet
NMLAN	I	Hold area	Number of lanes on freeway link, L
NODE	I	Hold area	Node locating off-ramp referenced by early warning sign
RNU	R	Dummy argument to RANDOM	

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/ LNKF
/A3/ SIGI
/A8/ IPACK
/A10/ LTRNV, MNUVR, RTRNV, THRUV
/FORINC/ IPOS1, IPOS2, LINK1, LINK2
/S2/ ITIMEF
/SIZE/ NLFO
/VBUF/ NF

5.13 Subroutine ALANE

Program Narrative

Purpose

ALANE identifies the origin lane of a lane-change maneuver made by a freeway vehicle.

Assumptions

None.

Approach

ALANE identifies the origin lane as the adjacent lane on either the left or right side of the lane currently occupied by the vehicle.

Synopsis

First a branch is taken depending on the direction of the change. If the change was to the left, the origin lane is the current lane minus 1 for lanes 2 through 5 or for auxiliary lane 7. For auxiliary lane 6, the origin lane will be the leftmost non-auxiliary freeway lane indicated by the variable NLANE (number of lanes on link). For lane 1 and auxiliary lane 8, the origin lanes will be auxiliary lanes 8 and 9, respectively. If the change was to the right, the origin lane is the current lane plus 1 for lanes 1 through NLANE or for auxiliary lane 6. For lane NLANE (leftmost non-auxiliary lane), the origin lane will be the auxiliary lane 6. For auxiliary lanes 8 and 9, the origin lanes will be 1 and 8, respectively.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IC	I	Hold Area	Origin lane
ILANE	I	Hold Area	Current lane

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIC	I	Hold Area	Output, origin lane
IILANE	I	Hold Area	Input, current lane
JDIR	I	Switch	Input, Code (0,1) if change was to (right, left)
NLANE	I	Hold Area	Input, number of thru lanes in link

Common Blocks and Variables Referenced

None.

5.14 Subroutine ASIG

Program Narrative

Purpose

To set the link-specific indications on all approach links to an actuated control node.

Assumptions

None.

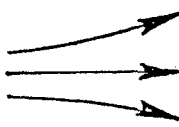
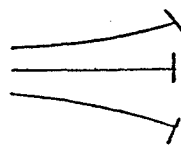
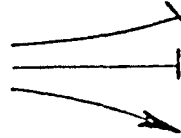
Approach

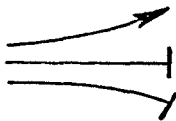
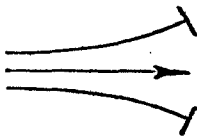
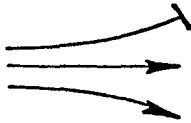
This routine sets the link-specific signal codes on all approach links of an actuated control node. The routine first identifies the signal codes that correspond to the current active phase on both rings and their respective overlapping phases, if any. Then the routine combines (forms a logical union of) these signal codes and subsequently calls a packing routine to store the new signal code. This process is repeated for all the approach links, L, at the actuated control node.

Synopsis

For a dual-ring controller and for a single-ring controller with overlapping phases, there may be two concurrent active phases. For this condition, it is necessary to identify the signal indication codes servicing each approach link that represent the logical union of these two concurrently active phases.

Consider the traffic movements and associated signal codes used in INTRAS:

Code	Movements	Code	Movements	Code	Movements
1		2		3	

<u>Code</u>	<u>Movements</u>	<u>Code</u>	<u>Movements</u>	<u>Code</u>	<u>Movements</u>
4		6	Red with diagonal green arrow	7	
8		9			

A code of 5 is a STOP-sign which is not applicable. The code of 0 (amber) is not presently applied but is included as a basis for possible future extension.

Suppose phases 1 and 5 are active concurrently. Assume that phase 1 (active alone) provides a signal code of 3 for a particular approach and that phase 5 provides a signal code of 7 for that approach. Hence, when both phases are active, the logical union of codes 3 and 7 is a code of 9 (see above sketches).

To implement the assignment of signal codes to each approach link every time there is a change in phase, a (symmetric) "truth matrix" is constructed. The elements of this matrix contain the signal codes for all possible logical unions of active phases:

		Signal Codes for Phases Active on Ring 1								
		1	3	4	8	9	7	2	0	6
Signal Codes for Phases Active on Ring 2	1	1	1	1	1	1	1	1	1	1
	3	1	3	8	8	9	9	3	3	1
	4	1	8	4	8	1	1	4	4	1
	8	1	8	8	8	1	1	8	8	1
	9	1	9	1	1	9	9	9	9	1
	7	1	9	1	1	9	7	7	7	1
	2	1	3	4	8	9	7	2	0	6
	0	1	3	4	8	9	7	0	0	6
	6	1	1	1	1	1	1	6	6	6

The above matrix is also applied when two overlapping phases on the same ring are active concurrently.

To use this matrix, it is necessary to identify the column (and row) associated, as shown above, and detailed below:

Signal Code	0	1	2	3	4	6	7	8	9
Column & Row	8	1	7	2	3	9	6	4	5

Returning to the example given above, the active Phase 1 on Ring 1 provides a signal code of 3 which maps into Column 2. The active phase, Phase 5 on Ring 2 provides a signal code of 7 which maps into Row 6. The union of these two codes, which is 9, is contained in cell (6,2) of the matrix.

This subroutine identifies the proper signal code for each approach link serviced by the actuated controller. The program loops to identify each approach link in turn. For each link, it identifies the active phase on Ring 1. If there is no active phase, or if the active phase is in the amber or red clearance interval, Column 7 is selected. Otherwise, the appropriate column is located, based upon the signal code extracted from the SIG array. (The program enters the SIG array with the known phase number.)

This procedure is repeated for the active phase on Ring 2 or for the slave phase overlapping on Ring 1. In either case, the appropriate row is located. With the row and column identified, the contents of the matrix cell thus defined is stored in the appropriate surface link array.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Approach to node N set in do-loop from 1 to 4
J	I	Hold Area	Code controlling link, L
JC	I	Hold Area	Value of row and column in MAT for signal code, J
JCOL	I	Hold Area	Column index in MAT array. Initialized to value 8
JROW	I	Hold Area	Row index in the MAT matrix
J4	I	Hold Area	Signal code
K	I	Hold Area	10 to the power of active phase on Ring 1 or raised to the power of active phase on Ring 2
L	I	Hold Area	Surface link number
M11	I	Hold Area	Active phase(s) on Ring 1
M12	I	Hold Area	Union of signal codes
M21	I	Hold Area	Active phase(s) on Ring 2
M22	I	Hold Area	Surrogate phase on Ring 2
NCODE	I	Hold Area	Phase on Ring 2, either amber or red clear or red revert
ND	I	Hold Area	Node number
NX	I	Switch	Code (1,2,3,4,5) if (normal, phase on Ring 1 is in amber, it is in red clear, phase on Ring 2 is in amber, in red clear)

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, internal sequenced actuated node number
I2	I	Hold Area	Input, Operation code if: 0, set link-specific signals normally 1, set the link-specific signals knowing that the phase on Ring 1 is in the amber interval 3, set the link-specific signals knowing that the phase on Ring 2 is in the amber interval

Common Blocks and Variables Referenced

/ACT2/ NAAFZ

/ACT12/ NASTAT

/A3/ SIGI

/ACT20/ SIGA

/ACT18/ ND

5.15 Subroutine BLOK

Program Narrative

Purpose

BLOK checks whether a given vehicle will be blocked by the next downstream incident, if any, and whether, given its new final speed, it can stop before reaching the incident.

Assumptions

BLOK assumes that subroutine CLOSE has been called previously to fill the LEND array with downstream incidents in each lane.

Approach

The LEND array is scanned to see if any incidents lie downstream of the given vehicle. The vehicle's new final speed is adjusted, if necessary, to allow the vehicle to stop before reaching the incident.

Synopsis

The original position of the given vehicle is computed. The LEND array is scanned to find the nearest downstream incident in the vehicle's lane. If there is none, or if the incident only causes rubbernecking, the new speed is acceptable and processing is finished. Likewise, if the vehicle, after attaining its new final speed, can stop before reaching the incident, the speed is acceptable. Otherwise, a quadratic equation must be solved to determine the adjusted speed and position to avoid collision (see Models and Algorithms Section).

If no speed adjustment will prevent collision, or if the vehicle must decelerate faster than the allowed maximum, emergency deceleration is applied.

If the adjusted speed requires the vehicle to stop during this time step, an error message is printed. If a collision cannot be avoided, a message is also printed.

If no change in new final speed was required, the adjusted final position is set equal to the given final position.

If the final speed was changed, the new position must be recomputed.

Models and Algorithms

The speed required to allow the vehicle to stop before colliding with the incident is found by solving simultaneous equations for speed:

$$P_2 - P_1 = \frac{v_1^2}{2a} \quad (\text{equation for stopping distance})$$

where P_2 is upstream end of incident

P_1 is final position of vehicle traveling at final speed, v_1

v_1 is final speed at end of time step

a is maximum allowable deceleration.

$$P_1 - P_0 = \frac{(v_1 + v_0)}{2} t$$

where P_0 is original position of vehicle

v_0 is original speed

t is time step.

Combining the two, we get the quadratic

$$2a(P_2 - P_0) = v_1^2 + v_1 at + v_0 at$$

And we solve to get v_1 . Then v_1 can be substituted to get P_1 .

Date, Programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
A	R	Hold Area	Second-order coefficient of quadratic equation
AMX	R	Hold Area	Maximum allowable deceleration from calibrated array
AMX1	R	Hold Area	Maximum allowable deceleration to avoid collision, equal to $AMX1/2$
B	R	Hold Area	First-order coefficient of quadratic equation
C	R	Hold Area	Constant coefficient of quadratic equation
D	R	Hold Area	Distance from predicted vehicle position to beginning of incident
DISC	R	Hold Area	Discriminant in quadratic equation solution
I	I	Do-Loop Index	Incident index, set in do-loop from 1 to 3
IDNI	I	Hold Area	Downstream end of incident
IOPOS	I	Hold Area	Position of vehicle at beginning of time step
IPN	I	Hold Area	Adjusted new position
IR	I	Hold Area	Temporary storage for unpacking LNEND array data
IRUB	I	Switch	(0,1) if there (is not, is) rubbernecking

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
ITT	I	Hold Area	Time interval during which emergency deceleration may be applied
IUPI	I	Hold Area	Upstream end of incident
MM	I	Constant	Set in data statement equal to 100000000
PI	R	Hold Area	Upstream end of incident
PN	R	Hold Area	Adjusted final position of vehicle

PO	R	Hold Area	Original position of vehicle
T	R	Hold Area	Time step
V	R	Hold Area	Given new speed
VN	R	Hold Area	Adjusted new speed of vehicle
VO	R	Hold Area	Original speed of vehicle

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
INS	I	Hold Area	Input, given final speed output, adjusted final speed
IOS	I	Hold Area	Input, original speed
IPOS	I	Hold Area	Input, original position
IT	I	Hold Area	Input, time step duration
IVT	I	Switch	Input, vehicle type
JPOS	I	Hold Area	Output, adjusted final position
LN	I	Hold Area	Input, lane number

Common Blocks and Variables Referenced

/ERTRAN/ IEPAR, IER, NPAR

/SIZE/ IS

/VEER/ LNEND

/A8/ MINAC

5.16 Subroutine CAL1

Program Narrative

Purpose

CAL1 implements clock time ramp metering.

Assumptions

CAL1 assumes, whenever it is called, that the node currently being processed is under clock time ramp metering control and that all necessary information to process the node under this type of control is stored in the IACPAR array. In addition, the assumption is made that the metering headway is always greater than a simulation time step.

Approach

CAL1 decides if it is time to update the signal code for an approach link to a node under ramp metering control by deciding if the metering headway has elapsed.

Synopsis

During the first time step of fill time CAL1 performs diagnostic tests on the ramp metering data stored in the IACPAR array (from card type 10) for the node being processed. Subsequent calls assume data in the IACPAR array is correct. For each pass through CAL1 during fill time the signal code facing the approach link to the actuated node is always set to the "Metered" (permissive) code. When fill time is just over and it is time for clock time ramp metering control to begin the signal code is set to the "Basic" (prohibitive) code. For all other passes the signal code is set to the "Metered" code only when the metering headway has elapsed. When a vehicle discharges onto the ramp entering the freeway, the signal code is set to the "Basic" code (see subroutine MOOV).

Models and Algorithms

The CALL algorithm sets the signal code facing the approach link to an actuated node at a frequency based on the metering headway.

Date, Programmer

January 1977. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBASIC	I	Hold Area	"Basic" signal code
IMOV	I	Hold Area	Movement code for which discharging vehicle triggers return to "Basic" signal code
K	I	Do-loop Index	Index of approaches to node under clock time metering control for approaches other than first. Set in do-loop from 2 to 4
LNUM	I	Hold Area	Number of approach link to actuated node
M	I	Hold Area	Link number unpacked from SIGI array for an approach to actuated node
MBEGIN	I	Hold Area	Time for onset of clock time metering, tenths-of-a-second
METER	I	Hold Area	"Metered" signal code
MHDY	I	Hold Area	Metering headway, tenths-of-a-second

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
N	I	Hold Area	Number of headways that have elapsed since clock time ramp metering began
NAC	I	Hold Area	Actuated node number corresponding to actual node number I

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Input, number of node to be processed under clock time ramp metering control

Common Blocks and Variables Referenced

/A3/ SIGI

/A6/ SIG, NPHS, SIGI

/CONTRL/ MXINIT

/ERTRAN/ IER, NPAR, IEPAR

/S2/ IDT, IDTS, ITIME

/SIZE/ IS

5.17 Subroutine CAL2

Program Narrative

Purpose

CAL2 implements demand/capacity ramp metering.

Assumptions

CAL2 assumes whenever it is called, that the node currently being processed is under demand/capacity metering control and that all necessary information to process the node under this type of control is stored in the IACPAR array.

Approach

CAL2 calculates headway for vehicles desiring to enter the freeway from the actual number of vehicles already on the freeway, determined over the last minute of simulation, and the freeway capacity.

Synopsis

During the first time-step of fill time, CAL2 performs diagnostic tests on the demand/capacity ramp metering data stored in the IACPAR array (from card type 10) for the node being processed. Subsequent calls assume data in IACPAR array is correct.

Headway is calculated from the actual volume of traffic on the freeway and the freeway capacity. During each call to CAL2, a count of vehicles crossing the detectors used to measure freeway performance is updated. When one minute has expired, the volume for the last minute of simulation is stored and the counter reset. The headway to be applied to vehicles desiring to enter the freeway is based on the difference of the freeway capacity and the last minute's volume. When the volume on the freeway is higher than the capacity specified, the headway is set to allow one vehicle on the freeway each minute.

The signal code facing the approach link to the actuated node is set to the "metered" code only when the headway has expired or when the headway is less than a time-step. When a vehicle discharges onto the ramp entering the freeway, the signal code is set to the "Basic" code (see subroutine MOOV).

Models and Algorithms

The CAL2 algorithm sets the signal code facing the approach link to an actuated node at a frequency based on the ratio of demand to capacity.

Date, Programmer

February 1977. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IALLOW	I	Hold Area	Number of vehicles/hr. to be allowed on freeway
IBASIC	I	Hold Area	"Basic" signal code
ICMC	I	Counter	Number of vehicles detected on freeway, is reset each minute
ICMT	I	Counter	Time, tenths-of-a-second. Is incremented each time-step and decremented after one minute has passed
ID	I	Hold Area	Detector number (index to DTCTR array)
IDTYP	I	Hold Area	Detector type
IHDY	I	Hold Area	Metering headway, tenths-of-a-second

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IMOV	I	Hold Area	Movement code for which discharging vehicle triggers return to "Basic" signal code
ISIG	I	Hold Area	Signal code currently facing approach link to actuated node
IT	I	Hold Area	Code (1,2,3) for (freeway, ramp, surface) link
ITOS	I	Hold Area	Time metered code was last set, tenths-of-a-second
K	I	General Purpose Do-loop index	
L	I	General Purpose Hold Area & Do-loop index	
LA	I	Index	IACPAR array index used when checking lane of detector
LANE	I	Hold Area	Lane containing detector to be used in measuring freeway volume
LD	I	Index	IACPAR array index used when checking detector position
LDPOS	I	Hold Area	Detector position, feet
LINDEX	I	Index	Index of LNKF array for detectorized freeway link
LKFNUM	I	Hold Area	Freeway link number of detectorized link
LMVOL	I	Hold Area	Volume for last minute of simulation

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LN	I	Hold Area	Lane in which detector is located
LNUM	I	Hold Area	Number of link approaching actuated node
LOC	I	Hold Area	Detector position, feet
LX	I	Index	Index for storing detector information in IACPAR array
M	I	Hold Area	Approach link number to actuated node

METER	I	Hold Area	"Metered" signal code
N	I	Index	Index of DTCTR array
NAC	I	Hold Area	Actuated node number corresponding to actual node number I
ND	I	Hold Area	Downstream node number of detectorized freeway link
NU	I	Hold Area	Upstream node number of detectorized freeway link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Input, number of node to be processed under demand/capacity ramp metering

Common Blocks and Variables Referenced

/A1/ LNKF

/A3/ SIGI

/A4/ DTCTR

/A6/ SIG, NPHS, SIGT

/CONTRL/ MXINIT

/ERTRAN/ IER, NPAR, IEPAR, IERCT

/S1/ MESERR

/S2/ IDT, IDTS, ITIME

/SIZE/ IS, NLFO

5.18 Subroutine CAL3

Program Narrative

Purpose

CAL3 implements speed control metering.

Assumptions

CAL3 assumes, whenever it is called, that the node currently being handled is to be processed under speed control metering and that all necessary information to process the node under this type of control is stored in the IACPAR array.

Approach

CAL3 evaluates the speed of vehicles already on the freeway and establishes a metering rate for those desiring to enter the freeway. The proper metering rate is determined from a table of speeds and metering rates (input on the type 10 card).

Synopsis

During the first time-step of fill time, CAL3 performs diagnostic tests on the speed control metering data stored in the IACPAR array (from card type 10) for the node being processed. Subsequent calls assume data in IACPAR array is correct.

Headway is determined from the average speed of traffic on the freeway. During each call to CAL3, a count of vehicles crossing the detectors used to measure freeway performance is updated and the speeds for these vehicles is summed. When one minute has expired, the average speed is calculated and the vehicle and speed counters are reset. The average speed value is compared to the table of speeds and metering headways (input by the user) to determine the proper headway. This headway is applied to vehicles desiring to enter the freeway during the current minute.

The signal code facing the approach link to the actuated node is set to the "metered code" only when the headway has expired or when the headway is less than a time-step. When a vehicle discharges onto the ramp entering the freeway, the signal code is set to the "Basic" code (see subroutine MOOV).

Models and Algorithms

The CAL3 algorithm sets the signal code facing the approach link to an actuated node at a frequency based on the average speed of freeway traffic.

Date, Programmer

March 1977. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBASIC	I	Hold Area	"Basic" signal code
ICMC	I	Counter	Number of vehicles detected on freeway. Reset each minute
ICMT	I	Counter	Time, tenths-of-a-second. Incremented each time-step and decremented after one minute has passed
ID	I	Hold Area	Detector number (index to DTCTR array)
IDTYP	I	Hold Area	Detector type
IHDY	I	Hold Area	Metering headway, tenths-of-a-second

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IMOV	I	Hold Area	Movement code for which discharging vehicle triggers return to "Basic" signal code
ISIG	I	Hold Area	Signal code currently facing approach link to actuated node
ISP(3)	I	Hold Area	Temporary storage array for speed values from speed and headway table
ISUM	I	Counter	Contains sum of speeds of vehicles detected on freeway, fps
IT	I	General Purpose Flag & Counter	
ITOS	I	Hold Area	Time metered code was last set, tenths-of-a-second
J	I	Do-loop Index	ISP array index, set in do-loop from 1,3
K	I	General Purpose Index & Do-loop Index	
L	I	Hold Area	Argument returned by subroutine FINDL (is not used by CAL3)
LANE	I	Hold Area	Lane containing detector to be used in measuring freeway volume and speeds
LDPOS	I	Hold Area	Detector position, feet

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LINDEX	I	Index	Index of LNKF array for detectorized freeway link or VF array index for a freeway vehicle
LKFNUM	I	Hold Area	Freeway link number of detectorized link
LMSPD	I	Hold Area	Average speed for last minute of simulation, fps
LN	I	Hold Area	Lane in which detector is located
LNUM	I	Hold Area	Number of link approaching actuated node
LOC	I	Hold Area	Detector position, feet
M	I	Hold Area	Approach link number to actuated node
METER	I	Hold Area	"Metered" signal code
N	I	Index	Index of DTCTR array
NAC	I	Hold Area	Actuated node number corresponding to actual node number I
ND	I	Hold Area	Downstream node number of detectorized freeway link
NU	I	Hold Area	Upstream node number of detectorized freeway link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Input, number of node to be processed under speed con- trol metering

Common Blocks and Variables Referenced

/A1/ LNKF

/A3/ SIGI

/A4/ DTCTR

/A6/ IACPAR, NPHS, SIG, SIGT

/CONTRL/ MXINIT

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/S1/ MESERR

/S2/ IDT, IDTS, ITIME

/SIZE/ IS, NLFO, NVFO

5.19 Subroutine CAL4

Program Narrative

Purpose

CAL4 implements gap acceptance merge control.

Assumptions

CAL4 assumes, whenever it is called, that the node currently being processed is under gap acceptance merge control and that all necessary information to process the node under this type of control is stored in the IACPAR array.

Approach

CAL4 stores all actuations and deactuations recorded by the coupled loop detector on the freeway. It uses this information to estimate when freeway vehicles (and therefore gaps) will arrive at the merge point. When a suitable gap is detected, a surface vehicle will be released onto the on-ramp so that it will merge smoothly into the gap.

Synopsis

During the first time-step of fill time, CAL4 performs diagnostic tests on the gap acceptance merge control data stored in the IACPAR array (from card type 10) for the node being processed. Subsequent calls assume data in IACPAR array is correct.

When CAL4 begins processing, it must update the information it keeps on freeway traffic traveling in the outside lane of the upstream freeway link (where the coupled loop is located). A list of actuations at the upstream loop of the coupled pair is kept. Each time a vehicle deactuates the upstream loop, its speed is calculated (from the deactuation and the previous actuation time and the loop length). This speed is used to update the prevailing speed which is an estimate of the speed of the freeway vehicles detected. Prevailing speed is calculated as follows:

$$S_2 = (3S_1 + S)/4$$

where

S = vehicle speed

S₁ = old prevailing speed (initial value is 75 fps)

S₂ = new prevailing speed.

When an actuation occurs at the downstream loop, an attempt will be made to determine which of the upstream actuations was made by the vehicle which has just crossed the downstream loop. A match occurs if

$$\frac{\text{downstream actuation}}{\text{time}} < \frac{\text{upstream actuation}}{\text{time}} + \frac{D}{.7S}$$

and

$$\frac{\text{downstream actuation}}{\text{time}} > \frac{\text{upstream actuation}}{\text{time}} + \frac{D}{1.3S}$$

where

D = detector spacing

S = prevailing speed.

A match indicates a vehicle has been identified. From the actuation times and the detector spacing, the vehicle's speed will be calculated and its arrival time at the merge point determined. If no match is found for the downstream actuation, the prevailing speed will be used to predict when the vehicle will arrive at the merge point (gore). This arrival time is calculated as follows:

$$T = T_1 + \frac{D}{S}$$

where

T₁ = time of actuation at downstream loop

D = distance from downstream loop to gore

S = vehicle speed (or prevailing speed)

T = time of arrival at gore.

These gore arrival times are stored in the IGT0A array.

The next task is to decide whether a surface vehicle should be released onto the on-ramp. First, the time it takes a queued vehicle to traverse the ramp link must be calculated. It is based on ramp length and free-flow speed. This time is added to the current time to get the time of arrival of the surface vehicle at the gore. If the vehicle will arrive at the gore during an acceptable gap (acceptable gap length is a user input) in freeway traffic, the signal code at the upstream end of the on-ramp will be set to the "metered" code. Otherwise, it will remain set to the "basic" code. After the vehicle discharges onto the ramp, the signal code is set to the "basic" code (see subroutine MOOV).

Models and Algorithms

The CAL4 algorithm sets the signal code facing the approach link to an actuated node to release vehicles so they may merge smoothly with other freeway traffic.

Date, Programmer

March 1977. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
FF	R	Hold Area	Ramp free-flow speed, fps
IBASIC	I	Hold Area	"Basic" signal code
ID	I	Hold Area	Detector number (index to DTCTR array)
IDIS	I	Hold Area	Distance from downstream de- tector of coupled pair to gore, feet

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDTYP	I	Hold Area	Detector type
IGTOA(10)	I	Hold Area	Array containing gore arrival times
IHAF	I	General Purpose Hold Area	
IL	I	Hold Area	Maximum time used when matching upstream actuations
IMOV	I	Hold Area	Movement code for which discharging vehicle triggers return to "Basic" signal code
IRAMPT	I	Hold Area	Time it takes a queued surface vehicle to traverse on-ramp
ISP	I	Hold Area	Speed of vehicle which has just deactuated the upstream loop
ISPEED	I	Hold Area	Speed of vehicle which has just actuated the downstream loop
IT	I	Hold Area	Code (1,2,3) for (freeway, ramp, surface) link
IU	I	Hold Area	Minimum time used when matching upstream actuations
IUPAC(4)	I	Hold Area	Array containing actuation times at upstream loop
IX	I	Hold Area	Code (0,1) if downstream actuation (was, was not) matched with an upstream actuation

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	General Purpose Hold Area and Do-loop Index	
JJ	I	General Purpose Hold Area	
JJJ	I	General Purpose Hold Area	
K	I	Do-loop Index or Hold Area	Set in do-loop from 2 to 4. Otherwise used to denote number of gore arrival times in IGTGA
L	I	General Purpose Do-loop Index and Hold Area	
LANE	I	Hold Area	Lane where coupled loop is located (from type 10 card)
LASTAC	I	Hold Area	Time of last actuation at upstream or downstream detector
LDPOS	I	Hold Area	Detector position, feet
LEN	I	Hold Area	Distance between loops, feet
LINDEX	I	Hold Area	Index of LNKF array
LKFNUM	I	Hold Area	Freeway link number of de- tectorized link
LN	I	Hold Area	Lane in which detector is located (from DTCTR array)
LNUM	I	Hold Area	Number of link approaching actuated node

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LTHRU	I	Hold Area	Freeway link number between detectorized link and gore
M	I	General Pur- pose Hold Area	
METER	I	Hold Area	"Metered" signal code
MM	I	General Pur- pose Hold Area	
N	I	Index	Index of DTCTR array for up- stream loop
NAC	I	Hold Area	Actuated node number corres- ponding to actual node number I
ND	I	Hold Area and Index	Downstream node number of freeway link or index of DTCTR array for downstream loop
NLANE	I	Hold Area	Number of lanes in freeway link
NODE	I	Hold Area	Downstream node of ramp link
NU	I	Hold Area	Upstream node number of free- way link
NUMR	I	Hold Area	Ramp link number
Y	R	Hold Area	Ramp length, feet

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Input, number of node to be processed under gap acceptance merge control

Common Blocks and Variables Referenced

/A1/ LNKF

/A2/ ENDNDS

/A3/ SIGI

/A4/ DTCTR

/A6/ IACPAR, NPHS, SIG, SIGT

/CONTRL/ ISCOT, MXINIT

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/S1/ MESERR

/S2/ IDT, IDTS, ITIME

/SIZE/ IS, NLFO, NLFR

5.20 Subroutine CAL5

Program Narrative

Purpose

CAL5 implements clock time diversion of freeway traffic.

Assumptions

CAL5 assumes, whenever it is called, that clock time diversion will be implemented at the node currently being processed and that all necessary information to process the node under this type of control is stored in the IACPAR array.

Approach

The percentage of vehicles to be diverted is applied to freeway vehicles (traveling through) on the link approaching the diversion node. All vehicles to be diverted are flagged to leave the freeway and follow the diversion path.

Synopsis

During the first time-step of fill time, CAL5 performs diagnostic tests on the clock time diversion data stored in the IACPAR array (from card type 10) for the node being processed. Subsequent calls assume data in IACPAR array is correct.

When processing begins, a test is made to determine if it is time to begin diverting vehicles. If not, control returns to the calling routine.

To determine the vehicles to be diverted, a scan is made through all freeway vehicles by means of a loop. Only those freeway vehicles on the link approaching the diversion node which are traveling through and which have just crossed the link boundary are considered for diversion. To determine if a vehicle has just crossed the link boundary, the distance travelled during the last time-step must be calculated. This distance, D , is calculated as follows:

$$D = (v-a/2) \cdot t$$

where

v = current speed of vehicle
 a = acceleration or deceleration of vehicle
 during time-step
 t = length of time-step.

The distance travelled is subtracted from the current position to determine the previous position. If the previous position indicates the vehicle was on another link, it will be considered for diversion. Otherwise, processing continues with the next vehicle.

To determine if a freeway vehicle satisfying the conditions described above will be diverted, a random number is generated. When the random number is less than or equal to the required percent, the following vehicle parameters will be set to cause the diversion: node number at which vehicle will be diverted (diversion node flag), flag indicating vehicle will leave freeway, lane vehicle prefers in order to exit, and desired lane switch code. The diversion node flag indicates which node the vehicle will be diverted at and, therefore, what path to follow since there may be more nodes employing clock time diversion.

In addition to setting up freeway vehicles for diversion, ramp and surface vehicles that have been diverted from the freeway must be moved along the diversion path. This is accomplished in a loop over all ramp vehicles and a loop over all surface vehicles. Only those vehicles with the diversion node flag set (equal to the node being processed) will have their turn codes adjusted to cause them to follow the diversion route. When a vehicle reaches the last link in the diversion sequence (on-ramp to a freeway or surface exit link), the diversion node flag (parameter 22 in VF array) will be cleared.

Models and Algorithms

The CAL5 algorithm diverts a portion of freeway traffic determined from a user-specified percent.

Date, Programmer

April 1977. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Dummy argument to RANDOM	
IHAF	I	General Purpose Hold Area	
INDEX	I	Hold Area	Index of IACPAR array which contains link number of last link in path
J	I	General Purpose Hold Area and Do-loop Index	
JHAF	I	General Purpose Hold Area	
K	I	General Purpose Hold Area and Do-loop Index	
KK	I	Hold Area	Parameter number of surface link array indicating type of receiving link. (Also used as dummy argument to RANDOM)
L	I	General Purpose Hold Area	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LINDEX	I	Hold Area	Index of LNKF array for link approaching diversion node or VF array index for freeway vehicle being processed
LINK	I	Hold Area	Link number of link in diversion path
LNUM	I	Hold Area	Number of link approaching diversion node
M	I	General Purpose Hold Area	
NAC	I	Hold Area	Actuated node number corresponding to actual node number I

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Input, number of node being processed under clock time diversion

Common Blocks and Variables Referenced

/A1/ LNKF, VF
 /A2/ ENDNDS
 /A3/ SIGI
 /A6/ IACPAR, SIG
 /CONTRL/ MXINIT
 /ERTRAN/ IEPAR, IER, IERCT, NPAR
 /S2/ IDT, IDTS, ITIME
 /SIZE/ IS, MXVFR, MXVRA, MXVSU, NLFO, NLFR, NLRA, NVFO

5.21 Subroutine CANCEL

Program Narrative

Purpose

To terminate the lane change book-keeping for a freeway vehicle.

Assumptions

CANCEL assumes the given link is a freeway link and that the NF array contains the vehicle parameters for the freeway vehicle being processed.

Approach

The two lane change indicators, freeway vehicle parameters 10 and 18, are set to indicate that no change is in process.

Synopsis

First CANCEL resets the lane change status flag (Parameter 10) to indicate that the vehicle's desired lane is or is not mandatory, and no change is currently in progress. Second, Parameter 18 is set to "3" to indicate that no lane change is in process or desired.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IC	I	Dummy Argument to DBUG	
IT	I	Switch	Code (0,1,2) if next link is a (freeway, ramp, surface) link
JDIR	I	Dummy Argument to DBUG	
M	I	Dummy Argument to DBUG	
ME	I	Dummy Argument to LASTV and NEXTLN	
MTYP	I	Switch	Code (0,1,2) if lane being processed is a (regular freeway lane or acceleration auxiliary, deceleration auxiliary, full auxiliary)
MUX	I	Dummy Argument to LASTV and NEXTLN	

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ILINK	I	Hold Area	Input, freeway link being processed
ILANE	I	Hold Area	Input, freeway lane being processed

Common Blocks and Variables Referenced

/VBUF/ NF

5.22 Subroutine CHANGE

Program Narrative

Purpose

To locate vehicles desiring to change lanes and CALL CHECK to perform the lane-change processing.

Assumptions

CHANGE assumes that the subject link parameters have been stored in the IPACK array.

Approach

CHANGE loops over vehicles in the specified link and locates vehicles desiring to change lanes. For each such vehicle subroutine CHECK is called to perform feasibility tests and implement the lane change.

Synopsis

CHANGE loops over all lanes on the specified link locating those vehicles which are farthest upstream.

The vehicle parameters are unpacked and it is determined whether the vehicle desires a change or is already changing. For those vehicles that wish to change lanes, the vehicles' target lane is identified. Routine CHECK is called to test whether a change is possible and, if so, implement the change. If the change is not possible, CHANGE determines whether a lane change in the other direction is acceptable. If so, CHECK is called again to attempt this change.

The next downstream vehicle is then processed until all vehicles on the subject link have been treated.

Models and Algorithms

None.

Date, Programmer

June 1976. David Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ICH	I	Switch	(0,1) Lane change (impossible, possible)
ID	I	Hold Area	Preferred lane number
ILANE	I	Hold Area	Lane number
ILINK	I	Hold Area	Link number
JHAF	I	Hold Area	Current speed of lead vehicle, fps
JPRILC	I	Hold Area	Lane number
LAUX	I	Hold Area	Number of auxiliary lanes processed
LINDEX	I	Hold Area	Number of vehicle in front of this vehicle in current lane
LY	I	Switch	Code (0,1) if vehicle will (turn, not turn)
M	I	Hold Area	Identification of the lane changing vehicle
MLAST	I	Hold Area	Upstream vehicle, in a given lane

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IL	I	Hold Area	Input, link number

Common Blocks and Variables Referenced

/A1/ VF

/A8/ IPACK

/SIZE/ NVFO

/VBUF/ NF

5.23 Subroutine CHECK

Program Narrative

Purpose

CHECK is called to determine if a specific lane-change can take place and, if so, to perform the lane-change and adjust bookkeeping accordingly.

Assumptions

It is assumed that the subject vehicle parameter vector has been stored in the NF array, and that IPACK contains the parameter vector for the link of current residence.

Approach

CHECK locates the vehicles in the target lane which lead and follow the subject vehicle. It is then determined whether a prohibitively hazardous situation would exist if the lane-change were to take place. If not, then the subject vehicle is moved to the new lane and all relevant bookkeeping is updated.

Synopsis

First, CHECK locates the leader and follower vehicles in the target lane. This is accomplished by calling LASTV to locate the vehicle furthest upstream in the target lane, and then, scanning down the link until a leader-follower pair is located. If both leader and follower are not located on the current link, then either LASTLK and/or NEXTLN are called to identify the upstream and/or downstream lane. The leader and follower parameters are then moved into the NFA and NFL arrays.

If the current lane is an acceleration lane, CHECK determines where it ends and sets parameters to indicate that there is no leader. If the subject vehicle is on a deceleration auxiliary lane, then CHECK locates the leader on this lane by calling routine UVR and sets parameters for subsequent processing.

Next the lane change desire is characterized as either "strong" or "weak." The following conditions would dictate a "strong" desire:

- Preferred lane is mandatory
- Vehicle is in an auxiliary
- Vehicle is destined for a downstream ramp and will force a lane-change
- Vehicle is travelling more than 20 fps below its desired speed.

For "weak" desired lane-changes the lane change will not take place if the new leader is not faster than the subject vehicle and is closer than 80 feet downstream.

The parameters are then set for the EMGNCY routine depending upon the strength of the desire to change lanes. EMGNCY is called to determine the required deceleration of the lane-changer to avoid colliding with the lead vehicle in the target lane. Then routine RISK is called to determine whether that required deceleration is acceptable to the candidate lane-changer. Parameters are next set for a call to EMGNCY where the trajectory of the follower in the target lane will be examined. EMGNCY is called to determine the required deceleration of the follower vehicle in the target lane in the event the lane change has taken place. RISK then determines whether this follower vehicle is willing to apply the required deceleration. If not, the lane change cannot take place.

The leader and follower identifiers in all relevant vehicle parameter vectors are updated when a lane change is implemented. The current lane for the subject vehicle is revised. The subject vehicles lane-change step counter is set indicating that a change is in progress. Also, the old follower vehicle is flagged to indicate the situation, so that it will not accelerate rapidly in order to close the gap to its new leader.

FRESET is called if the lane-changed has changed the identity of the last vehicle in either the original or target lanes.

Models and Algorithms

None.

Date, Programmer

June 1976. David Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold area	Acceleration of vehicle, fps
ICHEK	I	Switch	Code (1,2,3,4) if (no follower, no leader, follower and leader exist - leader on downstream link, leader and follower exist - both on subject link) in de- sired lane
ID	I	Hold area	Desired lane
IHAF	I	General purpose Hold area	
ILANE	I	Hold area	Lane vehicle M occupies (same as IILANE)
ILINK	I	Hold area	Link vehicle M occupies (same as IILINK)
ITY	I	Hold area	Type of link downstream of link ILINK and lane ID
IUPA	I	Switch	Code (0,1) if some element of NFA (has not, has) been revised
IUPL	I	Switch	Code (0,1) if some element of NFL (has not, has) been revised
IV	I	Hold Area	Vehicle type code
JAUX	I	Switch	Code (0,1,2) if vehicle MLAST is (not in an auxiliary, in auxiliary lane 1, in auxiliary lane 2)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JD	I	Hold area	Driver characteristic
JJ	I	Hold area	Grade code for subject link
JV	I	Hold area	Vehicle type code
KAC	I	Hold area	Acceleration of vehicle M during time step, tenths-of-a- foot/second/second
KACA	I	Hold area	Assumed deceleration of new leader for purposes of check- ing emergency stopping, tenths- of-a-foot/second/second
KACCL	I	Hold area	Maximum allowable acceleration for lane change, tenths-of-a- foot/second/second
KCARL	I	Hold area	Car length, feet
KDVA	I	Hold area	Vehicle free-flow speed, fps
KEACC	I	Hold area	Minimum acceleration, tenths- of-a-foot/second/second
KK	I	Hold area	MAXAC array index indicating speed
KKQ	I	Dummy argu- ments to RANDOM	
KLAST	I	Dummy argu- ments to LASTV	
KODE	I	Switch	Code (0,1) if there (is, is not) a follower vehicle in the tar- get lane
KOFF	I	Switch	Code (0,1) if change is (op- tional, mandatory)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KQ	I	Hold area	Random number
KTA	I	Hold area	Duration of lane change maneuver, tenths-of-a-second
KTB	I	Hold area	Duration of lane change maneuver, tenths-of-a-second
KV	I	Hold area	Vehicle speed, fps
KVC	I	Hold area	Projected speed at end of lane change, fps
KXA	I	Hold area	Distance of upstream vehicle on ramp from upstream end of ILINK, feet
LASL	I	Hold area	Link upstream of desired lane in subject link
LD	I	Hold area	Driver characteristic of ve- hicle M
LEX	I	Hold area	Position of vehicle M, feet
LINDEX	I	Hold area	Index of LNKF array
LKTYP	I	Hold area	Type of link upstream of link subject link
LNG	I	Hold area	Length of subject link, feet
LPREV	I	Hold area	Lane upstream of desired lane
LTHRU	I	Hold area	Lane downstream of desired lane
LTYPE	I	Switch	Code (0,1,2) if desired lane is (not an auxiliary, auxiliary lane 1, auxiliary lane 2)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LV	I	Hold area	Vehicle type code
M	I	Hold area	Subject vehicle, i.e., vehicle desiring to change lanes (same as MM)
MA	I	Hold area	Leader of subject vehicle in desired lane
MAUX	I	Dummy argu- ment to LASTV	
MEXL	I	Dummy argu- ment to NEXTLN	
MIA	I	Hold area	Leader of vehicle M in origin lane (same as MMIA)
MINF	I	Hold area	Minimum acceptable acceleration, tenths-of-a-foot/second/second
ML	I	Hold area	Vehicle in back of vehicle M in desired lane
MLAST	I	Hold area	Upstream vehicle in desired lane
MLT	I	Hold area	Upstream vehicle in lane feeding desired lane
MOF	I	Hold area	Old follower of vehicle M in origin lane
MOL	I	Hold area	Old leader of vehicle M in origin lane
MQQ	I	Hold area	Old leader of vehicle M in origin lane of link ILINK
NAUX	I	Dummy argu- ment to LASTV	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NEXL	I	Hold area	Link downstream of link I LINK and desired lane
NKPP	I	Switch	Code (0,1,2) if lane downstream of desired lane is (not an auxiliary, auxiliary lane 1, auxiliary lane 2)
NPPP	I	Switch	Code (0,1,2) if desired lane is (an acceleration, a deceleration, a full) auxiliary
NTHRU	I	Dummy argument to NEXTLN	
NTYP	I	Dummy argument to LASTV	
RNU	R	Dummy argument to RANDOM	

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
MM	I	Hold area	Input, vehicle changing lanes
IILANE	I	Hold area	Input, lane vehicle MM occupies
IILINK	I	Hold area	Input, link vehicle MM occupies
IID	I	Hold area	Input, number of desired lane
ICH	I	Switch	Output, code (0,1) if lane change is (impossible, possi- ble)
MMIA	I	Hold area	Input, leader of vehicle MM in original lane

Common Blocks and Variables Referenced

/A1/ LNKF, VF

/A8/ CLEN, IPACK, JERK, MAXAC, MINAC

/FPAR/ INCH, INRAMP, IRF, LAG, LAGB, NCH

/SIZE/ NLFO, NVFO

/VBUF/ NF, NFA, NFL, NRA, NRL

5.24 Subroutine CHOOZ

Program Narrative

Purpose

CHOOZ is called under three conditions: to determine if a lane change will occur, to determine if a driver will yield the right of way, or to determine if a driver will be leaving a freeway link.

Assumptions

None.

Approach

A flag is set to indicate if the event will or will not occur.

Synopsis

First a random number is generated. The random number is then compared to the probability of the event occurring (input to the subroutine). If the event occurs, the flag is set to 1. Otherwise, it is 0.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
M	I	Hold Area	Random number
N	I	Hold Area	Argument returned by subroutine RANDOM (is not used by CHOOZ)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
R	R	Hold Area	Argument returned by subroutine RANDOM (is not used by CHOOZ)

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IR	I	Hold Area	Input, probability of event occurring
IE	I	Hold Area	Output, flag - Code (0,1) if event (will not, will) occur

Common Blocks and Variables Referenced

None.

5.25 Subroutine CLNUP

Program Narrative

Purpose

CLNUP performs all necessary bookkeeping functions at the end of each time step which were not performed by other routines.

Assumptions

None.

Approach

CLNUP loops over all vehicle types and all vehicles of each type, performing required functions. It does the same for all incidents, all detectors, and all vehicles.

Synopsis

First, the process code is set for any dummy vehicles. Then CLNUP loops over all vehicle types and over all vehicles of each type. As each vehicle is encountered, the number of vehicles on the link is incremented. If the vehicle has not been processed this time step, a message is printed and the simulation is aborted. (This condition indicates a program error requiring debugging.) Otherwise, the process code is cleared and stored in the vehicle array. Then the total number of vehicles on each link is stored. (Note that links are processed in groups of a maximum of 115 because of size limitations on the MXI buffer which counts the number of vehicles on each link.)

Next, the incident array is examined, and completion codes are set for all those incidents which are over at this time.

Then, all non-plot, non-doppler detectors are examined. If the vehicle number is negative, this implies that the detector changed status this time step, and the vehicle number is made positive. Otherwise, the time for

which the current status has prevailed is incremented by the surface time step.

Finally, if a fuel tape is to be written, the trajectory data must be written out to the tape. CLNUP loops over all vehicles of each type, fills the fuel buffer with packed trajectory words and, as the buffer is filled, writes it out to the tape.

Models and Algorithms

None

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modifciation</u>
I	I	Do-loop Index	Vehicle type index, set in do-loop from 1 to 3
IA	I	Hold Area	Vehicle acceleration
IHAF	I	Hold Area	Temporary storage for packing and unpacking vehicle and link parameters
II	I	Do-loop Index	Vehicle index, set in do-loop from 1 to MX
IVT	I	Hold Area	Vehicle type code
J	I	Do-loop Index	Vehicle index set in do-loop from 1 to J3
JBASE	I	Hold Area	Offset to first link in group being processed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
JDT	I	Switch	Detector type, (0,>0) for (doppler, non-doppler)
JPT	I	Switch	(0,1) for (plot, non-plot) detector
JVACCP	I	Counter	Number of vehicle trajectory words
J1	I	Hold Area	Pointer to first link being processed
J2	I	Hold Area	Pointer to last link being processed
J3	I	Hold Area	Number of vehicles of type being processed
K	I	Hold Area	Index to MXI array - number of vehicles on link
KK	I	Switch	(0,1) for (acceleration, de- celeration)
L	I	Hold Area	Link number
MX	I	Hold Area	Number of freeway, ramp, or sur- face vehicles

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/ DTCTR, IDET

/A2/ ENDNDS

/S2/ IDT, IDTS, ITIME, ITIMEF, NIDT

/ERTRAN/ IEPAR, IER, NPAR

/CONTRL/ IFN, IFORM, ILEN

/A5/ INCID, NINCI

/SIZE/ IS, MXLFR, MXVFR, NLFR, NLRA

/CFUEL/ JFUEL, NU23

/NRGY/ JVACC

/VBUF/ NF

5.26 Subroutine CLOSE

Program Narrative

Purpose

CLOSE determines which lanes of a given link are closed due to incidents occurring at the current time or to the end of auxiliary lanes.

Assumptions

The given link number is assumed to be valid, and the link type is assumed to be on-ramp or freeway.

Approach

Both the given link and the downstream receiving link are checked for closed lanes (if the given link is a ramp, only the downstream link is checked). The beginning and end (if any) of closures in the lane are stored in the LNEND array.

Synopsis

CLOSE loops over the given link (unless it is a ramp link) and the downstream receiving link, processing each as follows: the pointer to the first incident is found, and each incident (if any) is examined. If the incident is not occurring at the current time, then it is ignored. Otherwise, the incident codes are unpacked, and the code for each lane of the given link (or, if the downstream link is being processed, each receiving lane of the given link) is checked; if the incident affects the lane, it must be stored in the LNEND array (a maximum of three incidents and/or lane end may be stored for each lane). If the incident occurs in an acceleration auxiliary lane, that lane is considered closed from the incident to the end of the lane. If it is a deceleration lane, it is considered closed from the beginning of the lane to the end of the incident. If a lane closes and

does not reopen, the end of the incident is defined as 9999.

The end of any auxiliary lanes as well as any through lane with no receiving lane are also stored as incidents.

When all incidents and lane ends in all lanes of the link and downstream link have been stored in the LNEND array, that array must be sorted such that the incidents for each lane are ordered from upstream to downstream locations.

If more than three incidents occur in one lane, an error message is printed and the simulation is aborted.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General-purpose do-loop index	
IAUX	I	Switch	(0,1) if end of auxiliary lane (has not, has) been processed
ICOD(9)	I	Hold Area	Unpacked incident codes for each lane
IDNA	I	Hold Area	Downstream end of auxil- iary lane location

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDNI	I	Hold Area	Location of downstream end of incident relative to given link
IDNIJ	I	Hold Area	Location of downstream end of incident relative to given link
II	I	Hold Area	IS array index
INC	I	Hold Area	Pointer to incident being examined
INCF	I	Hold Area	Pointer to first incident on link
IR	I	Hold Area	Temporary storage for unpacking incident codes
ITIM	I	Hold Area	Current surface or freeway simulation time
ITP	I	Switch	(0,1,2) for (accel, decel, accel + decel) auxiliary lane
IUPA	I	Hold Area	Location of upstream end of auxiliary lane
IUP1	I	Hold Area	Location of upstream end of incident
IUP2	I	Hold Area	Location of upstream end of 2nd incident to be sorted
J	I	Do-loop index	(1,2) if (given, downstream) link being processed
JJ	I	Hold Area	Auxiliary lane code index

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JLEN	I	Hold Area	Length of upstream link, if processing downstream link
JPACK(8)	I	Hold Area	Temporary storage of downstream link parameters
J1	I	Switch	(1,2) if given link is (freeway, ramp)
K	I	Do-loop index	Lane number, set in do-loop from 1 to 9
LD	I	Hold Area	Thru receiving link number
LENGTH	I	Hold Area	Upstream
LKK	I	Hold Area	Link number of link being processed
LK1	I	Hold Area	Link number of downstream link
LL	I	Hold Area	Temporary storage for sorting LNEND array
LN	I	Do-loop index	Lane number, set in do-loop from 1 to 9
LNCT(9)	I	Counters	Number of incidents and/or lane endings in each lane
LNI	I	Hold Area	Number of incidents or lane endings in given lane
LN11	I	Hold Area	LNI-1
LNN	I	Hold Area	Lane number of lane being processed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LN1	I	Hold Area	Downstream receiving lane
LTP1	I	Hold Area	Downstream receiving link type
MM	I	Constant	Set equal to 100000000 in data statement
M9	I	Constant	Set equal to 99999999 in data statement
NAUX1	I	Hold Area	First auxiliary lane code, if any
NAUX2	I	Hold Area	Second auxiliary lane code, if any
NLANE	I	Hold Area	Number of thru lanes in link
NXTYP	I	Hold Area	Link type of downstream receiving link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LK	I	Hold Area	Input, link number
LKTYP	I	Switch	Input, (0,1) for (free-way, ramp) link

Common Blocks and Variables Referenced

/CONTRL/IBUF,NEXCAL

/ERTRAN/IEPAR,IER,NPAR

/A5/INCID, NINCI

/A8/IPACK

/SIZE/IS

/S2/ITIME,ITIMEF

/VEER/LNEND

5.27 Subroutine COLECT

Program Narrative

Purpose

COLECT serves two distinct functions. The first is to accumulate statistics for the freeway data stations. The second function of this routine is to process all vehicle detectors to determine actuation times.

Assumptions

None.

Approach

The routine looks at a vehicle's initial and final position and determines whether a freeway data station or a vehicle detector has been crossed.

Synopsis

Each time a freeway vehicle is moved, this routine is called. The routine determines whether a vehicle has crossed a freeway data station during the current time-step. If so, the appropriate counters and data accumulation arrays are updated. Both lane change and link change situations are handled. It is possible for more than one freeway data station to be updated.

After the freeway data stations have been handled, both the link and lane the vehicle resides in are identified. All surveillance detectors in the given lane are identified and if any of them have been crossed by the vehicle, the time of actuation, and type of pulse (on, off), is determined. If a doppler detector has been actuated, the speed of the vehicle at the instant of actuation is computed. The data thus generated is transferred to a peripheral file for use off-line and the appropriate DTCTR array elements are updated.

Models and Algorithms

None.

Date, Programmer

January 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Hold Area	Vehicle acceleration
AJ	R	Hold Area	Freeway data station temporary storage
DELS	R	Hold Area	Distance travelled during time-step
DELT	R	Hold Area	Time to end of current time-step
HEADWY	R	Hold Area	Vehicle headway
I	I	Switch	I = (0,1) if we are handling (upstream, downstream) link with freeway data stations
IAUXY	I	Hold Area	Auxiliary lane number
ICASE	I	Switch	Detector actuation type (0,1,2,3,4) for (none, actuation, actuation/deactuation, deactuation, doppler actuation)
ICURST	I	Hold Area	Duration of current detector status
IDELT	I	Hold Area	Time to end of current time-step
IDP	I	Hold Area	Detector type
IDST	I	Hold Area	Distance
IDS1	I	Hold Area	Position of start of time-step

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDS2	I	Hold Area	Position at end of time-step
IDTY	I	Hold Area	Detector type
IGTCD	I	Switch	Determines section of code to return to after branch
IHFLAG	I	Hold Area	Freeway data station common storage
IIV	I	Hold Area	Last vehicle over detector
ILO	I	Hold Area	Data station number
IOCP	I	Hold Area	Detector status flag
IOFF	I	Switch	(0,1) to handle (actuation, actuation/deactuation)
IPRVST	I	Hold Area	Detector previous status time
IS2	I	Hold Area	Vehicle position after time- step
ITA	I	Hold Area	Time to detector actuation
ITD	I	Hold Area	Time to deactuation
ITLAST	I	Hold Area	Time of last actuation
IUPER	I	Hold Area	Distance from front of vehicle to rear of detector
J	I	Hold Area	General purpose index
KONOFF	I	Switch	Detector on/off flag
KTI	I	Hold Area	Detector time of actuation
LA	I	Hold Area	Detector lane
LANEN	I	Hold Area	Vehicle lane

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LEN	I	Hold Area	Vehicle length
LEND	I	Hold Area	Detector loop length
LINKN	I	Hold Area	Link vehicle is on
ND	I	Hold Area	Detector number
NUDE	I	Hold Area	Detector number
RADIC	I	Hold Area	Temporary storage
V1	R	Hold Area	Speed at start of time-step
V2	R	Hold Area	Speed at end of time-step

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/ NLV

/A4/ DTCTR, IDET

/A11/ ISTATN, LSTATN

/FORINC/ ICUR, IPOS1, IPOS2, ISPD1,
ISPD2, ISTART, ISTEP, IVNUM,
LANE1, LANE2, LENVH, LINK1, LINK2

/FREDAT/ FRCONT, FRFLAG, FRHDWY, FRSPED,
FRTIM

/SIZE/ NLFO

5.28 Subroutine CONSOL

Program Narrative

Purpose

CONSOL directs the processing of freeway traffic.

Assumptions

CONSOL assumes the given link is a freeway link.

Approach

CONSOL calls the proper routines to move freeway vehicles and bring in new vehicles from an on-ramp.

Synopsis

First, FMOVE is called to move freeway vehicles in the given lane of the given link. Then the number of the upstream vehicle in the given lane is updated in the freeway link array and ONRMP is called. After ONRMP brings in any new ramp vehicles desiring to enter the freeway, the upstream vehicle parameter is again updated.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
J	I	Hold Area	Index of LNKF array for up-stream vehicle parameter
L	I	Hold Area	Lane being processed
LINDEX	I	Hold Area	Index of LNKF array for link being processed
LK	I	Hold Area	Freeway link being processed

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IN	I	Hold Area	Input, lane number or code for auxiliary lanes being processed
ILINK	I	Hold Area	Input, number of freeway link being processed
JJ	I	Hold Area	Input, lane number or code (6,7) to indicate (first, second) auxiliary lanes

Common Blocks and Variables Referenced

/A1/ LNKF

/A8/ IPACK

/SIZE/ NLFO

5.29 Subroutine DECACT

Program Narrative

Purpose

To determine whether any phase now in green should enter clearance (amber) interval.

Assumptions

Assumes DETECT routine has calculated all the information pertaining to detectors.

Approach

This routine uses the tentative termination code set earlier by routine UPACT to determine whether any phase now in green should enter the clearance (amber) interval. For a single ring controller, the routine will move the phase in question into the amber interval if there is an outstanding phase call or the rest-in-red option is specified. For a dual ring controller, the decision to change the status of any phase(s) from green to amber interval depends on the position of the phase(s) in question with respect to the barrier of the controller and the current outstanding calls for competing phases.

Synopsis

In the case of a single ring controller, the currently active phase which has been flagged with the tentative termination code will remain active if there is no other phase call or the rest-in-red option is not specified.

In the case of a dual ring controller, the routine makes a distinction between whether there are one or two phases flagged with the tentative termination code. When one phase is flagged with the tentative termination code, the phase will enter the amber interval only if the phase in question is not at a barrier and its adjacent phase has been called. When two phases are flagged with the

tentative termination code, the decision to change the status of either one or both phases from green to amber is quite complicated. The following is a list of conditions that the routine has to identify before making the above mentioned decision:

- 1) Phase on ring one at barrier.
- 2) Phase on ring two at barrier.
- 3) Phase on ring one not at barrier.
- 4) Phase on ring two not at barrier.
- 5) Phase called adjacent to the phase in question on ring one.
- 6) Phase called adjacent to the phase in question on ring two.
- 7) Phase called not adjacent to the phase in question on ring one.
- 8) Phase called not adjacent to the phase in question on ring two.
- 9) No phase called on ring one.
- 10) No phase called on ring two.
- 11) Rest-in-red option specified.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Index, loops from 1 to 2
IX	I	Hold Area	Ring with active phase
IR	I	Hold Area	Ring where phase is terminating
MP1	I	Hold Area	Phase if no other phase is called and not rest-in-red

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>

MR2	I	Hold Area	"Slave" phase that overlaps the indicated parent phase

Arguments Passed To Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>

Il	I	Hold Area	Input, actuated node number

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT8/ NACDS

/ACT11/ NADEC

5.30 Subroutine DETECT

Program Narrative

Purpose

To determine if a vehicle has travelled over an assumed detector and to determine time of occurrence and vehicular speed.

Assumption

There is a detector at a constant distance from the stop line.

Approach

This routine determines whether a vehicle has travelled over an assumed detector during the current time-step. If so, the routine calculates, to the nearest tenth-of-a-second, the time of activation and the vehicle speed. These data are stored in the NEWDET array for further processing.

Synopsis

DETECT loops over all links serviced by an actuated controller. For each link, it loops over phases and detectors. The lane on which the vehicles are traveling and the lane on which the assumed detectors are placed are compared. If the lane checks out, the distance travelled by the vehicles and the distance where the assumed detectors are placed are compared. If the vehicle has crossed the assumed detector, the initial position of the vehicle is calculated. Otherwise, the routine examines the next vehicle. If the position of the detector is greater than the initial position and less than or equal to the final position, an activation has occurred. The speed and time are calculated and stored in an array NEWDET. After looping through all the detectors, all the phases and all the links, control returns to the calling routine.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
CURVEL	R	Hold Area	Square root of velocity squared
DELT	R	Hold Area	Calculated time of vehicle actuation
FINSPD	R	Hold Area	Final free-flow speed
I	I	Do-Loop Index	Index, set in do-loop from 1,2
ICODE	I	Hold Area	Code (0,1,2) if vehicle (is not, is, is because of cycle failure) in queue
ICRVEL	I	Hold Area	Rounded to INTEGER value of CRVEL
IDELT	I	Hold Area	Rounded to INTEGER value of DELT
IDIST	I	Hold Area	Distance of vehicle from upstream node
II	I	Do-loop Index	Index, set in a do-loop from 1 to 7
ILAN	I	Hold Area	Lane number
ILINK	I	Hold Area	Link number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ISP	I	Hold Area	Free-flow speed when vehicle is first in queue
ISTDET	I	Hold Area	Distance of the assumed detectors from stop line. Initialized at 60 feet
IV	I	Hold Area	Vehicle number
PB	R	Hold Area	Real value of distance of vehicle from upstream node
PS	R	Hold Area	Real value of distance of assumed detector from upstream node
S	R	Hold Area	Distance from initial to final position
SL	R	Hold Area	Distance travelled by the vehicle when activation occurred

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
II	I	Hold Area	Input, internal sequenced actuated node number

Common Blocks and Variables Referenced

/ERTRAN/ IER, NPAR, IEPAR
 /ACT0/ ISTDET
 /ACT1/ NAFZ
 /ACT7/ NALNK
 /ACT4/ NADET
 /ACT15/ NEWDET
 /ACT10/ NACT
 /VBUF/ NF

5.31 Subroutine DETSW

Program Narrative

Purpose

To implement detector switching at a given actuated control node.

Assumptions

Assumes DETECT routine has provided pertinent information regarding detectors.

Approach

None.

Synopsis

This routine implements the detector switching logic for a dual ring controller. The routine first identifies the phase which has attained its maximum green, then switches the detector actuations associated with this phase to the active phase on the other ring. This feature is only applicable when the two active phases are both at a barrier. This feature is designed to extend the current active phases until the minimum gap is exceeded on all approaches serviced by both phases (or until both phases have attained or exceeded, their maximum duration).

The detector switching is not applicable if any one of the following conditions is met:

- Single ring controller;
- One of the active phases is not at a barrier;
- One, or both, of the active phases is actuated by presence detectors;
- Neither of the active phases has attained maximum green or maximum extended green.

The routine, therefore, must first test all of the above conditions; and, when none of them are met, it will then assign the smaller value of the inter-arrival gap to both phases on both rings.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Smallest value of inter-arrival gap
J	I	Hold Area	End of yield interval
K	I	Hold Area	Phase actuation code
L	I	Hold Area	Active phase

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, node identifier

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT8/ NACDS

5.32 Subroutine EMGNCY

Program Narrative

Purpose

EMGNCY calculates the required deceleration of the follower vehicle in order to avoid collision.

Assumptions

None.

Approach

The lead vehicle is processed to a designated point in time and its speed, position and acceleration are used to compute the required deceleration of the follower vehicle.

Synopsis

First, EMGNCY artificially processes the leader of the pair of vehicles to a designated point in time. Then, an emergency constraint function is used to compute the acceleration that the follower vehicle must undergo so that it will be safely behind the new leader at the end of the lane change time predicted. A reference for the constraint function is provided below.

Models and Algorithms

See the INTRAS Final Report, Volume 1, Appendix B, for a complete description of the derivation of the emergency constraints.

Date, Programmer

January 1977. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KACCA	I	Hold Area	Current acceleration of lead vehicle in fpss $\times$ 10
KCARL	I	Hold Area	Length of the leading vehicle, feet
KDVA	I	Hold Area	Desired free-flow speed of lead vehicle, fps
KEACC	I	Hold Area	Maximum deceleration of following vehicle in fpss $\times$ 10
KTA	I	Hold Area	Remainder of lane change time that will occur after this time-step is over, tenths-of-a-second
KTB	I	Hold Area	Time period until current lane change is complete, tenths-of-a-second
KVA	I	Hold Area	Current speed of lead vehicle, fps
KVA1	I	Hold Area	Speed of lead vehicle at designated point in time, fps
KVB	I	Hold Area	Current speed of following vehicle, fps
KXA	I	Hold Area	Distance of lead vehicle from upstream node, feet
KXA1	I	Hold Area	Distance of lead vehicle from upstream node at designated point in time, feet
KXB	I	Hold Area	Distance of follower vehicle from upstream node, feet

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JKACCA	I	Hold Area	Input, current acceleration of lead vehicle in $\text{fpss} \times 10$
JKCARL	I	Hold Area	Input, car length of lead vehicle with no spacing buffer, feet
JKDVA	I	Hold Area	Input, desired free-flow speed of lead vehicle, fps
JKEACC	I	Hold Area	Input, maximum deceleration of following vehicle in $\text{fpss} \times 10$
JKTA	I	Hold Area	Input, remainder of lane change time that will occur after this time-step is over, tenths-of-a-second
JKTB	I	Hold Area	Input, time period until current lane change is complete, tenths-of-a-second
JKVA	I	Hold Area	Input, current speed of lead vehicle, fps
JKVB	I	Hold Area	Input, current speed of follower vehicle, fps
JKXA	I	Hold Area	Input, distance of lead vehicle from upstream node, feet
JKXB	I	Hold Area	Input, distance of follower vehicle from upstream node in feet
KACC1	I	Hold Area	Output, minimum required deceleration to avoid collision in $\text{fpss} \times 10$

Common Blocks and Variables Referenced

/FPAR/ KET, LAG

5.33 Subroutine FGNRAT

Program Narrative

Purpose

FGNRAT generates new vehicles on freeway entry links.

Assumptions

FGNRAT assumes that current information is contained in the IPACK and NF arrays.

Approach

FGNRAT generates vehicles on an entry link which feeds the first link of the freeway. The vehicle characteristics are randomly generated and this information is stored in the freeway vehicle array.

Synopsis

First, the routine checks if there is any vehicle in the specified lane of the generation link. If there is, routine FMOVE is called to move the vehicle ahead. The identification number of the entry vehicle is checked. If equal to zero, FGNRAT would return to FMAIN without generating any new vehicles. Otherwise, a new vehicle is generated. The vehicle parameters are initialized, to be packed later. The emission headway and the percent of vehicle types is determined for this lane. The vehicle type is generated by calling routine TYPE, with the distribution of vehicle type as input. The driver type is determined stochastically and the desired free-flow speed is obtained on the basis of lane position, vehicle type and driver type. If there is a leader vehicle in the entry link, the new vehicle position must take the leader into consideration. Finally, the vehicle data is packed, and the new vehicle becomes the farthest upstream vehicle in the lane.

Models and Algorithms

None.

Date, Programmer

January 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Dummy Argument to RANDOM	
AMEAN	R	Hold Area	Mean time headway
IFEN	I	Hold Area	Identification number of entry vehicle
IHDY	I	Hold Area	Space headway
IJ	I	Do-loop Index	General purpose do-loop index
IJTYP	I	Dummy Argument to NEXTLN and LASTV	
ILANE	I	Hold Area	Entry lane number
ILINK	I	Hold Area	Entry link number
JAUX	I	Dummy Argument to LASTV and DEBUG	
JITYP	I	Dummy Argument to LASTV	
JJ	I	Hold Area	Percentage of vehicles assigned to each lane

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JSUM	I	Hold Area	Percentage of vehicles assigned to lanes 2 to 5
JTEMP (10)	I	Hold Area	Percentage of vehicle type assigned to each lane
JTPCT	I	Hold Area	Percentage of commercial vehicles assigned to each lane
JV	I	Switch	Vehicle type code
KK	I	Hold Area	Percentage of vehicle type assigned to each lane
KLNPCT (5)	I	Hold Area	Percentage of vehicles assigned to each lane
KPOS	I	Hold Area	Length of freeway link, feet
KSUM	I	Hold Area	Percentage of vehicle type assigned to lanes 2 to 5
KVTPCT (5)	I	Hold Area	Percentage of vehicle type assigned to each lane
LAUX	I	Dummy Argument to LASTV	
LD	I	Switch	Driver type code
LINDEX	I	Hold Area	Number of vehicle in front of this vehicle in current lane
LTHRU	I	Dummy Argument to NEXTLN and LASTV	
LTOTC	I	Hold Area	Percentage of vehicles assigned to lanes 3 to 5

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LTOTP	I	Hold Area	Percentage of vehicle type assigned to lanes 1 and 2
LTYPE	I	Dummy Argument to LASTV and DBUG	
LV	I	Switch	Vehicle type code
M	I	Hold Area	Empty slot in freeway vehicle array
MA	I	Hold Area	Leader of this vehicle in current lane
NEXL	I	Dummy Argument to NEXTLN and LASTV	
NLAN	I	Hold Area	Total number of lanes minus 1
NLAST	I	Hold Area	Last vehicle to enter link
VOL	R	Hold Area	Traffic volume

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIFEN	I	Hold Area	Input, identification number of entry vehicle
IILANE	I	Hold Area	Input, entry lane number
IILINK	I	Hold Area	Input, entry link number

Common Blocks and Variables Referenced

/A1/ VF

/A7/ SNODE

/A8/ CLEN, IPACK, ITPCT, LNMNSP, VFHF

/A10/ MNUVR

/FPAR/ MINHDY

/SIZE/ NVFO

/VBUF/ NF

5.34 Subroutine FILL

Program Narrative

Purpose

FILL fills the link environment with information pertaining to a given approach or receiving link.

Assumptions

FILL assumes that SELEN has determined the geometric location of the given approach or receiving link.

Approach

The given link is processed according to whether it is a surface receiving link or approach link, or a ramp receiving link or approach link. In each case the link environment is filled appropriately.

Synopsis

Processing is as follows, depending on whether the given link is a receiving or approach link, ramp or surface link:

(a) Surface exit link:

The number of lanes is stored. All other items remain zero.

(b) Surface receiving link:

The link parameter vector is unpacked, and the receiving link environment parameters are set, as well as parameters pertaining to the farthest upstream vehicle in each lane.

(c) Surface approach link

The link parameter vector is unpacked, and the receiving link environment parameters,

as well as parameters pertaining to the farthest downstream vehicle in each lane, are set.

(d) Ramp receiving link

Same as for surface receiving link except that ramp link and vehicle arrays are accessed.

(e) Ramp approach link

Same as for surface approach link except that ramp link and vehicle arrays are accessed.

(f) Freeway receiving link:

NEXTLN must be called to determine each freeway receiving lane of the link whose environment is being established, so that the information for the farthest upstream vehicle in each lane can be unpacked.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews

Program Variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Switch	(1,2,3) for (left, thru, right) approach or receiving link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IFG	I	Switch	(1,2) for (receiving, approach) link
IP1	I	Hold area	Percent of volume making first ramp movement
J	I	Hold area	Argument to NEXTLN, not used
K	I	General- purpose do-loop index	
KM	I	Hold area	Parameter number of farthest upstream vehicle in each lane
L	I	Hold area	Number of lanes in on-ramp
LK	I	Hold area	Link number
LTEM	I	Hold area	Farthest upstream vehicle in lane
LTEMP	I	Hold area	Temporary storage for unpacking vehicle parameters
LTHRU	I	Hold area	Receiving lane number
LTYPE	I	Switch	Link type code, (0,1,2) for (freeway, ramp, surface)
M	I	Hold area	Farthest downstream vehicle
N	I	Hold area	Argument to LASTV, not used
NEXL	I	Hold area	Receiving link number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NTYP	I	Switch	2 for freeway exit or non-existent link, 0 otherwise

Arguments passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IFLAG	I	Switch	Input, (1,2) for (receiving, approach) link
INDEX	I	Switch	Input, (1,2,3) for (left, thru, right) approach or receiving link
LKNUM	I	Hold area	Input, link number
LLTYPE	I	Switch	Input, (2,1,0) for (surface, ramp, freeway) link

Common blocks and variables referenced

/A8/CLEN,IPACK

/A2/IVFEET

/LENVI/LACA,LACR,LCCR,LDHR,LDTR,LFFR,
LFTR,LINK,LLLA,LLNA,LLNR,LLPA,LLTR,
LMHR,LMTR,LPOA,LPOR,LPRA,LPRR,LRPA,
LRTR,LSCA,LSHR,LSPA,LSPR,LSTR,LTCA,
LTTR,LUVR,LVLR

/SRFLAG/LTFLAG

/VBUF/NF

/SIZE/NLFR,NLRA

5.35 Subroutine FMAIN

Program Narrative

Purpose

FMAIN controls the simulation of freeway traffic and the emission of new vehicles on freeway links.

Assumptions

None.

Approach

FMAIN loops over all freeway links and lanes. Calls the proper routines to move and emit freeway vehicles. It also initiates the lane change logic on internal freeway links.

Synopsis

FMAIN processes freeway links in downstream to upstream order. For each link the vector of link parameters is unpacked and tests for lane ends (for auxiliary lanes). Lane blockages are made by calling CLOSE. For each internal freeway link, CONSOL is called to move vehicles. If the link is a freeway entry, FGNRAT is called to emit vehicles. After traffic has been moved on all freeway links, freeway vehicles desiring a lane change are processed (by calling CHANGE).

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Freeway link index. Set in do-loop from 1 to NTOTL
IFEN	I	Hold Area	Entry link number, or 0 if excess vehicles have been generated on the entry link being processed
IJ	I	Do-loop Index	Entry link number. Set in do-loop from 1 to NS
ILINK	I	Hold Area	Freeway link number for link currently being processed
IN	I	General purpose Hold Area and Do-loop Index	Lane number of lane currently being processed
IREQ	I	Hold Area	Number of vehicles that should have been generated
IX	I	Hold Area	Lane number of lane being processed
LINDEX	I	Hold Area	Index of LNKF array
NLANE	I	Hold Area	Number of lanes on freeway link being processed

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/ LNKF

/A2/ ENDNDS, NTOTL

/A7/ NS, SNODE

/A8/ IPACK

/A10/ MNUVR, THRUV

/CONTRL/ MAXSTS

/ERTRAN/ IEPAR, IER, NPAR

/FPAR/ INCH, INRAMP, IORD, IRF, NCH

/ONVEN/ NORG

/RANK/ IFRANK

/S2/ IDT, IDTF, IDTS, IFDT, ITIME,
ITIMEF, NIDT

5.36 Subroutine FMOVE

Program Narrative

Purpose

FMOVE moves and determines current lane-change desire for all vehicles in a given lane of a given freeway link.

Assumptions

FMOVE assumes that the subject link parameters have been stored in the IPACK array.

Approach

Vehicles are processed starting from the downstream most vehicle and proceeding in sequence upstream. The car following and emergency deceleration laws are applied to determine inhibitions due to the downstream traffic. Current impedances, desired-lane and random processes govern the generation of lane-change desire.

Synopsis

Initially, FMOVE sets a code to discourage lane changing in the direction of an auxiliary lane if one exists. Then, the next link and lane are located by calling NEXTLN. The first vehicle in this lane is determined by calling FRSTV. FMOVE unpacks that vehicle vector by calling UNPAK and increments the flag that limits the jerk of this vehicle in the event the downstream vehicle has changed lanes.

If the subject vehicle is on a lane that leads to a ramp and has no leader, then FMOVE locates last vehicle on ramp using routine LASTV and unpacks that vehicle's parameters using routine UVR.

If the subject vehicle is in an auxiliary lane, has no lead vehicle and wishes to remain on the freeway, FMOVE sets up a dummy lead vehicle that is motionless at the point where the auxiliary ends.

If the subject vehicle has a leader and wishes to continue on the freeway, FMOVE sets vehicle parameter values and calls BLOK to determine whether there is a rubbernecking condition downstream.

If the subject vehicle is changing lanes and is headed for an off-ramp, FMOVE calculates the vehicle's desired acceleration bypassing the car-following law.

For all vehicles with a leader, FMOVE identifies the coefficient in the car-following law that reflects whether rubbernecking is present. Then, the car-following law is applied to obtain the acceleration. The computed acceleration is then revised as required by other constraints. Routine EMGNCY is called to calculate the emergency acceleration and adjust the calculated value accordingly. The computed values of speed and acceleration are adjusted in accord with the grade and his desired speed.

Then FMOVE identifies whether there is a blockage downstream which will affect the vehicle's trajectory this time-step by calling routine BLOK. If so, the vehicle's acceleration is adjusted. The current link statistics are adjusted to reflect the movement of this vehicle.

Parameters are then primed for the purpose of providing input information on the subject vehicles movement for routines COLECT and ADVANC. COLECT is then called to process detector actuations and ADVANC to respond to advance warning signs. The updated vehicle parameters are then stored. Routine CANCEL is called for the purpose of resetting the codes if the subject vehicle has completed the lane change.

FMOVE then checks whether a lane change is mandatory and sets the desired code accordingly. If a lane-change is not mandatory, FMOVE determines whether it is desirable because of impedance or because vehicle wishes to return to preferred lane. The desired code is set accordingly. It is then determined whether a desired lane-change can be accomplished (i.e., it is not physically unreasonable because of geometric conditions). Finally, if this vehicle

crosses into the downstream link, routine LCROSS is called to adjust the bookkeeping to reflect that it is in a new link.

The vehicle vector is then stored and the trailing vehicle is located as the next one to be processed if within this link. The new vehicle data is unpacked and the above procedure is repeated until all vehicles in the subject lane and link have been processed.

Models and Algorithms

Car-Following

The procedure for normal car following is composed of the following three steps:

(1) $KACC = (KXA - KX - KCARL - (KDT + KF) * KV) / (KDT * KDT + 2 * KF * KDT)$
where:

- KDT = the scanning time interval
- KACC = the computed acceleration of the follower during the scanning interval
- KX = the coordinate position of the follower at the beginning of the scanning interval
- KV = the speed of the follower at the beginning of the scanning interval
- KXA = the coordinate position of the leader at the end of the scanning interval; if the leader is the downstream termination of the auxiliary lane, KXA=0
- KCARL = the car length of the leader; if the leader is the downstream termination of the auxiliary lane, KCARL = 0
- KF = the K factor of the follower (a factor which determines the desired spacing between the leader and the follower). KF is based upon, driver type dependent, built-in values in the FOLK array. It may be modified by a rubbernecking factor in the presence of incidents.

(2) Compute the minimum deceleration required by the follower, KACCL, to avoid a collision with its leader at the end of the next scanning interval. The routine EMGNCY is called to perform this using KX, KXA, KVA (the speed of the leader at the end of the specified scanning interval), KTA (the time duration from the end of the specified scanning interval) and KEACC (the emerging deceleration) as inputs.

(3) The computed KACC should satisfy the following conditions:

- a) KACC should be less than KACCL
- b) KACC should be less than the maximum acceleration of the follower
- c) KACC should be greater than the maximum deceleration of the follower
- d) the change of acceleration rate should be less than the maximum jerk of the follower.

If any of these conditions is violated, the proper boundary value is assumed to be the acceleration of the follower.

Date, Programmer

June 1976. David Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General purpose Do-loop index and hold area	
IA	I	Dummy argument to LASTV	
ID	I	Hold area	Type of link downstream of link being processed
IDES	I	Switch	Code (0,1) if event (will not, will) occur
IHAF	I	Hold area	Number of vehicles discharged from ramp feeding link ILINK
IHICAP	I	Switch	Code (0,1) if on-ramp (does not, does) join subject link at upstream or downstream end. Used to temporarily in- crease capacity upstream of a ramp merge.
II	I	General purpose Hold area	
ILAN	I	Hold area	Lane in downstream link fed by ILANE
ILANE	I	Hold area	Lane being processed
ILEPT	I	Hold area	Extreme left lane or lane to left of preferred lane
ILINK	I	Hold area	Link being processed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IRIGHT	I	Hold area	Extreme right lane or lane to right of preferred lane
ISHY	I	Switch	Code (0,1,2) if vehicle changing lanes (will not shy away from new lane, will shy away from left auxiliary, will shy away from right auxiliary)
ISHYNO	I	Switch	Code (0,1) indicating (weak, strong) lane change motivation
ITOPSP	I	Hold area	Minimum of vehicle free-flow speed and limiting speed for this vehicle type
IUPA	I	Switch	Code (0,1) if some element of NFA (has not, has) been revised. Therefore, NFA must be packed
J	I	Hold area	Lane code (same as JJ)
JCO	I	Hold area	Sensitivity factor which determines car following distance
JHAF	I	Hold area	Number of lanes, in ramp link feeding link ILINK
JJJ	I	General purpose Hold area	
JLANE	I	Hold area	Number of adjacent lane
JV	I	Hold area	Vehicle type code
K	I	General purpose Do-loop index and hold area	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KAC	I	Hold area	Acceleration of vehicle M
KACA	I	Hold area	Acceleration of vehicle MA
KACC	I	Hold area	Acceleration of subject, vehicle hundredths-of-a-foot/ sec ²
KACCL	I	Hold area	Required emergency accelera- tion
KACCO	I	Hold area	Original acceleration of sub- ject vehicle before checking for collision against end of acceleration-deceleration lane and incidents
KALC	I	Hold area	Limiting value of accelera- tion due to jerk considera- tions
KCARL	I	Hold area	Car length of leader vehicle
KDVA	I	Hold area	Desired free-flow speed, fps
KES	I	Hold area	Time during which new accel- eration will prevail, tenths- of-a-second
KET	I	Hold area	Same as KES but in hundredths- of-a-second
KF	I	Hold area	Car following spacing fac- tor, hundredths-of-foot/sec.
KINS	I	Hold area	New speed of subject vehicle corresponding to KIPOS
KIOS	I	Hold area	Original speed of vehicle M, fps

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KIPOS	I	Hold area	Final position of subject vehicle
KIT	I	Hold area	Processing time during this time step (from original to final position)
KJPOS	I	Hold area	Adjusted final position of vehicle M
KKQ	I	General purpose Hold area	
KQ	I	Hold area	Random number
KRUB	I	Hold area	Rubbernecking factor
KT	I	Hold area	Processing time of subject vehicle, tenths-of-a-second
KTa	I	Hold area	Remainder of lane change time that will occur after this time step is over, tenths-of-a-second
KTb	I	Hold area	Time period until current lane change is complete, tenths-of-a-second
KV	I	Hold area	Speed of subject vehicle before processing, fps
KVA	I	Hold area	Speed of leader at the end of this time step, fps
KVB	I	Hold area	Speed of subject vehicle at end of processing, tenths-of-a-foot/sec
KX	I	Hold area	Position of subject vehicle before processing, feet

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KXA	I	Hold area	Position of leader at the end of this time step, feet
KXB	I	Hold area	Position of subject vehicle after processing, tenths-of-a-foot
L	I	Hold area	Upstream or downstream node number of link ILINK
LANE3	I	Hold area	Temporary storage during definition of LANE1 and LANE2
LANE4	I	Hold area	Temporary storage during definition of LANE1 and LANE2
LASL	I	Hold area	Link upstream of subject link
LD	I	Hold area	Driver characteristic of subject vehicle
LEXIT	I	Hold Area	Code (0,1) if vehicle (has not, has) left freeway
LGRAD	I	Hold area	Grade code of subject link
LKLEEV	I	Hold Area	Code (0,1) if vehicle (does not, does) change links during this time step
LKTYP	I	Hold Area	Type of link upstream of subject link
LPREV	I	Dummy argument to LASTLK	
LSPD	I	Hold area	MAXAC array index indicating speed range
LTHRU	I	Hold area	Lane downstream of subject lane

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LTYPE	I	Switch	Code (0,1,2,3) if subject lane is an (acceleration, deceleration, both acceleration-deceleration, regular freeway) lane
LV	I	Hold area	Subject vehicle type
M	I	Hold area	Subject vehicle
MA	I	Hold area	Leader vehicle
MLAST	I	Hold Area	Upstream vehicle
MQ	I	Hold Area	Last vehicle processed in subject lane
NEXL	I	Hold Area	Link downstream of subject link
NKTYP	I	Hold Area	Type of link downstream of subject link
NN	I	Do-loop index	Set in loop from 1 to 25
NTYPE	I	Dummy argument to LASTV	
RNU	R	Dummy argument to RANDOM	

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IIILANE	I	Hold area	Input, freeway lane number, being processed or auxiliary lane number
IIILINK	I	Hold area	Input, freeway link number being processed
IJ	I	Hold area	Input, lane code (1 thru 7) for lane being processed. If less than 6 code equals actual lane number. Code equals 6 for first auxiliary lane and 7 for second auxiliary lane

Common Blocks and Variables Referenced

/A2/ ENDNDS, IVFEET

/A3/ SIGI

/A3/ CLEN, CSTDCL, FOLK, IPACK, JERK, LIMSPD,
MAXAC, MINAC

/CONTRL/ ISTFLG

/FORINC/ ICUR, IPOS1, IPOS2, ISPD1, ISPD2,
ISTART, ISTEP, IVNUM, LANE1, LANE2,
LENVH, LINK1, LINK2, NCHNG

/FPAR/ INCH, IP, KET, LAG, LAGA, LAGB,
MENAC, NCH

/S2/ IDT, IDTF, IDTS, ITIME, ITIMEF

/VBUF/ NF, NFA, NRA

5.37 Subroutine FRESET

Program Narrative

Purpose

FRESET is called to update the parameter in the freeway or ramp link array defining the upstream vehicle in a given lane of a given link.

Assumptions

None.

Approach

FRESET determines the index of the array being updated and stores the upstream vehicle number in the location indicated by the index.

Synopsis

This routine may update either the freeway link array or the ramp link array. The link type (input to the subroutine) indicates which will be changed. Once the link type is known, the index of the array location to be changed is determined and the upstream vehicle number stored. The routine also updates a buffer containing the unpacked link array.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LINDEX	I	Hold Area	Index of LNKF array
MQ	I	Hold Area	Upstream vehicle number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IK	I	Hold Area	Input, link number
IL	I	Hold Area	Input, lane number
M	I	Hold Area	Input, upstream vehicle number
IKTYP	I	Hold Area	Input, link type
LBUF(45)	I	Hold Area	Input, unpacked link array

Common Blocks and Variables Referenced

/A1/ LNKF

/SIZE/ NLFO

5.38 Subroutine FZCL

Program Narrative

Purpose

To determine whether there is a phase called at an actuated controller.

Assumptions

Assumes DETECT routine has provided pertinent information.

Approach

This routine scans through all the detector actuation counters that service a given phase. If there has been an actuation on one of the detectors servicing the phase following the current active phase, that phase number will be returned through the argument list. If none of the detectors have been actuated, the routine will then repeat the above process for the next phase. After all the available phases on both rings have been searched, the routine will return a zero in the argument list if no actuation was found.

Synopsis

The routine will set the phase in question to be the phase that was called if the phase is non-actuated or if the recall switch is on or if there is an actuation indicated by one of the detectors which is servicing this phase. The routine repeats all the above testing for the next phase until one of the above conditions is met. When all phases on both rings have been tested, the routine will return a zero in the argument list to indicate that there is no outstanding phase called if none of the above conditions are satisfied.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Detector index, set in do-loop from 1,8
IR	I	Hold Area	Identification of the ring
J	I	Hold Area	Other Ring

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, phase identifier
I3	I	Hold Area	Output, active phase number

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT6/ NACNT

/ACT8/ NACDS

5.39 Subroutine GETCD

Program Narrative

Purpose

GETCD assigns a new turn code to a vehicle as it enters a new link.

Assumptions

GETCD assumes that the link and vehicle environment areas are filled with data pertaining to the vehicle in question.

Approach

A turn code is assigned by comparing a uniform random number with turn percentages; it may be changed if too many vehicles have already been assigned to the turn code.

Synopsis

RANDOM is called to generate a uniform random number. This is compared with the turning percentages on the receiving link to yield a new turn code. Before it is accepted, however, the actual number of total vehicles on the link making the desired turn is compared with the desired number. If it exceeds the desired number by more than two vehicles, one of the two remaining turn codes--the one whose actual number of vehicles is lower than desired -- is assigned to the vehicle.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDUM	I	Hold area	Uniform random number from RANDOM, not used
ITC	I	Switch	(1,2,3) for (left, thru, right) movement
I1	I	Hold Area	Index of first turn total to be compared
I2	I	Hold Area	Index of second turn total to be compared
L	I	Hold Area	Index to MNUVR
LL	I	Hold Area	Link which just received given vehicle
NR	I	Hold Area	Uniform random number between 1 and 100
P	R	Hold Area	Percentage of vehicles making each turn
RDUM	R	Hold Area	Uniform random number from RANDOM, not used
VA(3)	R	Hold Area	Each turn movement from this link
VI(3)	R	Hold Area	Desired numbers of vehicles making each turn movement from this link
VTOT	R	Hold Area	Total number of vehicles discharged from this link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NTC	I	Switch	Output, new turn code assigned to vehicle, (0,1,2) for (left turn, right)

Common Blocks and Variables Referenced

/A8/IPACK

/LENVI/IRALL,LANE,LINK,LLTR,LRTR,LTSURF,
LTTR

/SRFLAG/LTFLAG

/A10/LTRNV,MNUVR,RTRNV,THRUV

/VBUF/NF

/SIZE/NLFR,NLRA

5.40 Subroutine GETUNV

Program Narrative

Purpose

GETUNV unpacks one vehicle parameter by calling the appropriate unpacking routine.

Assumptions

GETUNV assumes all arguments passed to it are valid.

Approach

None.

Synopsis

If the link in question is a surface link, UVS is called to unpack the desired parameter; otherwise, UVR is called.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IPP	I	Hold Area	Unpacked parameter

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IN	I	Hold Area	Input, parameter number
IP2	I	Hold Area	Output, parameter value
IV	I	Hold Area	Input, vehicle number

Common Blocks and Variables Referenced

/SRFLAG/ LTFLAG

5.41 Subroutine GOQ

Program Narrative

Purpose

GOQ checks spillback conditions on cross-links and also looks for acceptable gaps in oncoming traffic from cross-links and opposed link. GOQ then decides whether or not the given vehicle can discharge.

Assumptions

GOQ assumes that the vehicle and link environment areas are filled with data pertaining to the vehicle in question.

Approach

First, spillback on cross-links is checked. Next, gaps in oncoming traffic from the opposed link are examined. Finally, gaps on the cross-links are checked. If at any point a test fails, the vehicle will not be allowed to discharge.

Synopsis

First, spillback on cross-links is checked (this check is skipped for right-turning vehicles). GOQ loops through the same code for both near and far receiving links, skipping the check for the far link for a left-turning vehicle (taken care of by TSTSAT) or for a thru vehicle not in the leftmost lane. Each lane of the cross-link is examined as follows: First, if the upstream vehicle in the receiving link is flagged as just having discharged from the given link, it is ignored. If there is no upstream vehicle, of course, there is no spillback. Otherwise, GOQ computes the time required for the given vehicle to reach the lane being checked while accelerating at its maximum allowable rate (if there are any intervening lanes to cross). Next, the position of the upstream vehicle in the receiving link lane must be computed at the beginning of the time step and also at the end of the time step plus the travel time just computed. If, at the end of this time, the upstream vehicle exceeds the maximum allowable spill-

back, the given vehicle will not discharge. If there is no spillback at the beginning of the time step, the lane is acceptable. If there is spillback at the beginning but not at the end, acceptability depends on the geometry: if the blocker is in any but the nearest lane of the near receiving link, it is acceptable. If it is in any but the nearest lane of the far receiving link, or if there is a near receiving link intervening, it is acceptable. That is, unless the spillback is extremely close to the given vehicle, it is assumed that it will clear before the vehicle has time to travel to the lane in question.

If all lanes have been found acceptable, GOQ proceeds to check the gaps in oncoming traffic from the opposed link (this section is skipped for thru vehicles and right-turners, or if there is no opposed link, or if a prohibitive signal faces the opposed link). GOQ loops over all lanes in the opposed link. If there is no thru vehicle in the lane, the lane is acceptable. If the opposed vehicle is in queue, but the subject vehicle is not, the lane is acceptable. If both are in queue and the subject vehicle is a ramp vehicle, the lane is acceptable. If the opposed link has already been processed (links are processed in order by number), it is acceptable.

Otherwise, for both in queue, the subject vehicle will discharge with .5 probability. If, on the other hand, the opposed vehicle is in motion, the standard approach is taken: the lag, or the time required for the opposed vehicle to travel to the leftmost lane of the far receiving link (into which the subject vehicle will presumably discharge), is computed. If it is greater than the minimum required gap (found in the calibrated array according to driver type), the lane is acceptable; otherwise, the vehicle is prevented from discharging.

If all lanes on the opposed link have acceptable gaps, GOQ goes on to check for acceptable gaps on the cross-links. (This check is skipped if the signal is anything but stop, yield or RTOR). If the sign is a yield sign, a lag time reduction factor is computed proportional to vehicle speed (the faster the vehicle is moving, the smaller the lag required for it to discharge).

If the gap has never been checked before for this vehicle, it will be checked now. If it was checked but it

is not time to check it yet, the vehicle will not discharge (see Models and Algorithms section). So, if the cross-links are to be checked, GOQ loops through the same code for both near and far cross-links, after computing the minimum acceptable gap (determined from a decile distribution combined with the yield sign reduction factor, if any). If the vehicle is a right-turner, only the near cross-link is checked. If there is no approach link or if a prohibitive signal is facing it, no further checking is necessary. The appropriate intersection width needed for lag computation and the appropriate lanes to be checked are determined. GOQ then loops over all the lanes. If the farthest downstream vehicle in the lane being checked is stopped, the lane is acceptable. If the vehicle is moving, the lag, or the time required for the vehicle to travel to the location of the subject vehicle is computed. If the gap is smaller than the minimum allowable gap, the vehicle cannot discharge. If all lanes on both cross-links have acceptable lags, the vehicle is allowed to discharge, as far as GOQ is concerned.

Models and Algorithms

In order to simulate a real-world situation for checking gaps on cross-links, a vehicle is not allowed to check at every time step. If, on the first check a gap is rejected, a counter is stored in the vehicle array (in place of free-flow speed which does not apply as long as the vehicle is in queue). This counter is decremented at each call to GOQ so that the vehicle can check the gaps once every twenty seconds. This prevents the vehicle from checking the same gaps over and over and finding them unacceptable each time. The counter is reset to the worst possible gap encountered on any subsequent pass so that time will be allowed for that approaching vehicle to go by.

Date, Programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
AA	R	Hold Area	Second-order coefficient in equation for distance traveled
BB	R	Hold Area	First-order coefficient as above
CC	R	Hold Area	Constant coefficient as above
DISC	R	Hold Area	Discriminant in solving quadratic equation
I	I	Do-Loop Index	(1,2) for (near, far) link being checked
IDC	I	Hold Area	Driver type code
IDUM	I	Hold Area	Uniform random number, output from RANDOM, not used
IGAPA	I	Hold Area	Minimum acceptable gap
IGC	I	Hold Area	Grade code index
II	I	Hold Area	Index to link environment data
IPOS1	I	Hold Area	Position of rear of blocker at beginning of time step
IPOS2	I	Hold Area	Position of rear of blocker at end of time step plus time required to cross intervening lanes
ISPX	I	Hold Area	Minimum downstream position of blocker to avoid spillback
IT	I	Hold Area	Time step

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IV	I	Hold Area	LLOR index of given lane
IVT	I	Hold Area	Vehicle type index
IX	I	Hold Area	LLOR index of leftmost lane in link
IVY	I	Counter	Time remaining until next cross-link check can be made
IL	I	Do-Loop Index	LADICH array index, set in do-loop from 1 to 6
K	I	Do-Loop Index	LLOR index, set in do-loop from 1 to 5
LAG	I	Hold Area	Lag of approach vehicle on cross-link
LL	I	Do-Loop Index	Lane index of lane being checked, set from 1 to LNU
LLL	I	Hold Area	Link environment lane index
LLN	I	Hold Area	Intersection width
LLO	I	Hold Area	Lane number, unpacked from LLOR array
LLX	I	Hold Area	Index to ITGAP array
LNU	I	Hold Area	Number of lanes on cross-link
LNUT	I	Hold Area	Number of thru lanes on cross-link
LT	I	Hold Area	Lag reduction factor for vehicle at yield sign
MY	I	Hold Area	Calibrated minimum lag minus reduction factor

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
NLC	I	Hold Area	Number of lanes vehicle must cross to reach spillback
NR	I	Hold Area	Uniform random number between 1 and 100
RDUM	R	Hold Area	Uniform random number, output from RANDOM, not used
T	R	Hold Area	Time required to cross NLC lanes

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
J	I	Switch	Output, (0,1) if vehicle (will, will not) discharge

Common Blocks and Variables Referenced

/S2/ IDTS, ITIME

/A8/ IGAP, IPACK, ITGAP, LSLAG, MAXAC

/VENVI/ JPFOL, JPLED, JSIGP, JSIGS, JVIDF, JVIDL

/LENVI/ LACR, LADICH, LANE, LINK, LLA, LLNA, LLNR, LLOR, LLPA, LPOA, LPOR, LPRA, LPRR, LSCA, LSPA, LSPR, LTCA, LUVR, LVL R

/SRFLAG/ LTFLAG

5.42 Subroutine GRNSIG

Program Narrative

Purpose

To update the initial, the added initial, and the green interval of an active phase.

Assumptions

None.

Approach

This routine updates the initial, the added initial, and the green intervals of an active phase. In the case of initial or added initial interval, the routine simply checks the appropriate interval timer and moves to the next interval if the maximum duration for that interval has been attained. In the case of green interval, the interval will be flagged to terminate when either the maximum green has been exceeded or the minimum inter-arrival gap has exceeded the current passage time.

Synopsis

The routine branches according to the status code that was passed through the argument list. For the initial and the added initial intervals, the routine will only check whether or not the duration of the respective interval has reached its specified maximum. If it has, then a proper adjustment on the status code will be made to reflect the beginning of a new interval. For the green interval, the current passage time is reduced by the specified rate until it reaches its minimum value. The routine will set the tentative termination flag if either the minimum inter-arrival gap has exceeded the current passage time or the maximum green has been attained. The only time that an absolute termination flag is set is when the active phase of a coordinated control reaches its force-off point.

Models and Algorithms

None.

Date, Programmer

May, 1977. Meena Magarajan

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
X	I	Hold Area	Passage time of active phase
EG	I	Hold Area	Minimum gap
KK	I	Hold Area	Smallest inter-arrival gap
LFF	L	Hold Area	Code if coordinated and phase is forced off
MTMR	I	Hold Area	Time for active indication if status code less than 6

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, ring identifier
I3	I	Hold Area	Input, coordination code
I4	I	Hold Area	Input, passage time

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT11/ NADEC

/ACT12/ NASTAT

5.43 Subroutine HDWY

Program Narrative

Purpose

HDWY computes the discharge headway for a given vehicle on a given link of a given link type.

Assumptions

HDWY assumes that the link environment area contains all data pertaining to the given link.

Approach

First, the mean queue discharge headway is found, and then it is adjusted according to the vehicle's original queue position and vehicle type, if necessary.

Synopsis

First, the driver type and vehicle type are unpacked. The mean queue discharge headway is found in the IPACK array, as well as the original queue position of the given vehicle. If the queue position is invalid (not between 1 and 4), then an error message is printed, and the simulation is aborted. The discharge headway is computed from the mean queue discharge headway multiplied by a calibrated distribution according to intersection type and driver type. If the vehicle was first in queue, this is the final result. If it was second or third in queue, the headway is adjusted further by an additive factor depending on original queue position and by a multiplicative factor depending on vehicle type. The headway is returned through the argument list.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Index to original queue position
IDT	I	Hold Area	Driver type code
IORQ	I	Hold Area	Original queue position
ITYPE	I	Hold Area	Intersection type code
IVT	I	Hold Area	Vehicle type code
MQDH	I	Hold Area	Mean queue discharge headway

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IHY	I	Hold Area	Output, discharge headway
LN	I	Hold Area	Input, lane number
LT	I	Switch	Input, (1,2) for (ramp, surface) link
MM	I	Hold Area	Input, vehicle number

Common blocks and variables referenced

/ERTRAN/IEPAR,IER,NPAR

/A8/IPACK,TLNK

/LENVI/LANE,LINK

5.44 Subroutine INCDAT

Program Narrative

Purpose

INCDAT accumulates detector pulses and outputs them to a detector data file.

Assumptions

The routine assumes that two peripheral files have been allocated. One is used for the on-line incident detection. The second is used for the INCES module.

Approach

As each detector pulse is generated in routine CSELECT, INCDAT is called. The detector number, type of pulse, and time of actuation are loaded into an output buffer. When the buffer is full, it is dumped to the peripheral data file.

Synopsis

On the first call to the routine, the peripheral data file is positioned. If there are existing data files on the disc, they are skipped and an inter-file code is written.

In addition to positioning the output file on the first call, the routine writes a block of header information. Network geometry, and run identification data are dumped.

On all calls to the routine with detector pulse data, the output buffer is loaded, counters are incremented, and, if the buffer is full it is dumped. A number of calls to this routine may be made without detector pulse data. These calls are used to empty the buffer at the end of subintervals and at the end of simulation.

Models and Algorithms

None.

Date, Programmer

December 1976. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Used in various loops.
IADD	I	Hold Area	Sum of the number of freeway, ramp and maximum number of sur- face links.
IBOUT	I	Array	Output buffer.
ICIN	I	Array	Output buffer, equivalenced to IBOUT.
IHOLD	I	Hold Area	Used in sort of detector actuations.
II	I	Hold Area	Link counter.
INMB	I	Hold Area	Used in sort of detector actuations.
IPACK2	I	Array	Buffer used to unpack link data.
JLD	I	Hold Area	Output buffer count.
K	I	Hold Area	Output buffer counter.
KLD	I	Hold Area	Output buffer counter
K1	I	Do-loop Index	Sort index

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K2	I	Hold Area	Used in sort
L	I	Do-loop Index	Loop to load secondary output buffer
LINKD	I	Array	Output buffer, equivalenced to IBOUT
NLNK	I	Hold Area	Link counter
NTA	I	Hold Area	Peripheral file number for INCES tape
NTEM	I	Hold Area	Peripheral file number for on-line tape

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NUDE	I	Hold Area	Input, detector number
KTI	I	Hold Area	Input, time of actuation in milliseconds
KONOFF	I	Hold Area	Input, type of pulse (9,1) for (off, on) pulse for loop detector. Speed at actuation in MPH*100 for doppler detectors

Common Blocks and Variables Referenced

```

/A2/ ENDNDS
/A4/ DTCTR,IDET,NDET
/CONTRL/ IDRUN,IFN,IFORM,IIDMOD,IIDOP,IINOT,ILEN,
        IOP,MXINIT
/CRASH/ KALL,LNKSTA,MAXSTA
/ERTRAN/ IER,IERCT
/FORINC/ ICALL,IINC,LINEON,NUMBER
/INCDMP/ KBOUT

```

/RANK/ IFRANK, NFENT
/S2/ IDTF, IDTS, ITIME, ITIMEF
/SIZE/ MXLFR, MXLRA, MXLSU, NLFR, NLSU

5.45 Subroutine LANES

Program Narrative

Purpose

LANES assigns a given vehicle to a new lane as it discharges onto a new link.

Assumptions

LANES assumes that the link and vehicle environment areas contain all information pertaining to the vehicle in question. LANES also assumes that it has already been determined that the vehicle will definitely discharge; therefore, an acceptable lane must be found.

Approach

The candidate lanes and their relative desirabilities are determined, depending on the vehicle's present and future turn codes. Then the candidate lanes are examined and the best one is chosen to be the new lane for the given vehicle.

Synopsis

First, GETCD is called to determine the vehicle's new turn code. Then the candidate receiving lanes must be determined and stored, in order of decreasing desirability, in the LNCHK array. For a left turning vehicle, the receiving lane is checked first, then the lanes to the left of the receiving lane, then those to the right. For a right turning vehicle, the receiving lane, and then lanes to the right are checked. For a thru vehicle, the receiving lane is checked first, and then the lanes immediately to the left and right, the order determined by the vehicle's new downstream turn code.

Then LANES loops through all entries in the LNCHK array, choosing the first acceptable lane as follows: if there is no upstream vehicle in the lane, then it is acceptable. Otherwise, the final position of the upstream vehicle is compared with the predicted final position of the subject vehicle; if they will not collide, then the lane is

acceptable. If the given vehicle can avoid collision by decelerating to a halt at a reasonable rate (7 fpss for thru and right, 14 fpss for left turners), then the lane is acceptable. If the lane is not acceptable, then LANES keeps track of the "best" lane found so far; that is, the one whose upstream vehicle is farthest downstream after emergency deceleration. If, when all lanes have been examined, none has been found to be truly acceptable, then a warning message is printed and the best available lane is assigned to the vehicle. (This condition probably would indicate a modelling error requiring further debugging.)

Models and Algorithms

None.

Date, Programmer

January, 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	LNCHK index, set in do-loop from 1 to LNX
IHM	I	Hold Area	Minimum vehicle horizon, set to dummy value
II	I	Hold Area	LLOR index of vehicle's current lane
IPOS	I	Hold Area	Final position of rear of up- stream vehicle in receiving lane
ITC	I	Switch	Present turn code, (0,1,2) for (left, thru, right) vehicle
IX	I	Hold Area	LLOR index of leftmost lane in current link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IZL	I	Hold Area	Leader position after emergency deceleration
IZLMAX	I	Hold Area	Position after emergency deceleration of leader in best lane so far
IZS	I	Hold Area	Vehicle position after reasonable deceleration to halt
J	I	Do-loop Index	Lane index, set from 1 to JJ
JDECL	I	Hold Area	Normal deceleration to avoid collision
JDECL2	I	Hold Area	JDECL*2
JJ	I	Hold Area	Number of receiving lanes next to left of vehicle
JPOS	I	Hold Area	Final position of subject vehicle on new link
JPOS0	I	Hold Area	Initial position of subject vehicle on new link
LBEST	I	Hold Area	Lane number of best lane so far
LN	I	Hold Area	Lane number of lane being checked
LNC	I	Hold Area	Lane number of best candidate lane
LNCHK(5)	I	Hold Area	Array containing candidate lanes for receiving lane
LNK	I	Hold Area	Number of lane entries in LNCHK array

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
NLANE	I	Hold Area	Number of thru lanes in receiving link
NTC	I	Hold Area	New turn code, determined by GETCD routine

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/S2/ IDTS, ITIME

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/A8/ IPACK

/VENVI/ JVFAC, JVFLN, JVFLT, JVFS

/LENVI/ LACR, LANE, LINK, LLNA, LLNR, LLOR,
LPOR, LPRR, LSPR, LTSURF, LUVR,
LVLR

/SRFLAG/ LTFLAG

/VBUF/ NF

5.46 Subroutine LCROSS

Program Narrative

Purpose

LCROSS is called whenever a vehicle leaves a freeway link to update the statistics of the links in question and the subject vehicle.

Assumptions

It is assumed that the subject vehicle parameter vector has been stored in the NF array, and that IPACK contains the parameter vector for the original link.

Approach

LCROSS locates any newly visible leader vehicles, sets link-specific vehicle parameters and then moves the vehicle to its new link position. Link statistics are updated for both original and final links.

Synopsis

For vehicles leaving a freeway link and entering an off-ramp, LCROSS creates a ramp vehicle vector, determines the new ramp vehicle number by calling FINDRV, and then calls OFFRMP to position the new ramp vehicle. The freeway vehicle is then deleted.

The link parameter vector for the new link is unpacked into the KROSBF array.

For vehicles entering a new freeway link, if there is no vehicle in the receiving lane of the new link, LCROSS looks downstream to the next receiving link. It, then, calls LASTV to identify the last vehicle in the receiving lane of that link as the subject vehicle's new leader. The subject vehicle is then placed on the next link and the link statistics are updated. FRESET is called to identify this vehicle as last in its new lane.

A turn code is assigned for this vehicle in its new link and then modified if necessary to satisfy the specified turning rates. The assigned turn code is checked to be sure that it does not violate barrier curbs. If the vehicle is to be a through vehicle in the new link, his desired lane is assigned to be compatible with his current lateral position and through movement.

For vehicles which will leave the freeway at the downstream end of the new link, the lane that feeds the exit ramp is determined and identified as the desired lane for this vehicle. The lane change code is set to "mandatory."

If the new link is an exit, then LCROSS clears the vehicle array, resets the statistics of the following vehicle on the old link and returns.

Models and Algorithms

None.

Date, Programmer

June 1976. David Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Set in do-loop from 1 to 25
IHAF	I	General purpose Hold area	
ILANE	I	Hold area	Original lane
ILINK	I	Hold area	Original link
IQA	I	Hold area	Desired lane
IQB	I	Hold area	Percent of vehicles desiring lane IQA
JDLANE (5,4,5)			Distribution of traffic over lanes when there is not a one to one correspondence between lanes in upstream and downstream links. JDLANE(I,J,K) contains XXXY where XXX is percentage of vehicles having a desired lane of I which shall be assigned to desired lane of Y. (J = number of lanes -1 in downstream link and K = number of lanes in upstream link.) For the case where there are K lanes in the upstream link and J+1 lanes in the downstream link remaining traffic having a desired lane of I in upstream link would be assigned to lane I+1 in downstream link.

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JHAF	I	General purpose Hold area	
JTC	I	Switch	Code (0,1) if vehicle (will not, will) turn at end of this link
K	I	Hold area	Number of lanes on subject link
KAUX	I	Dummy argu- ment to LASTV	
KDEFA	I	Hold area	Actual percent of vehicles that have turned
KGO	I	Switch	Code (0,1) if vehicle may go (right, left)
KHAF	I	Switch	Code (0,1,2) if auxiliary lane 2 is (acceleration, decelera- tion, both)
KKQ	I	General purpose Hold area	
KQ	I	Hold area	Random number
KROSBF (47)	I	Hold area	Array of link parameters for new link
KTOT	I	Hold area	Total number of vehicles enter- ing downstream link during sub- interval
KTURN	I	Hold area	Number of vehicles that have turned so far

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IILINK	I	Hold area	Input, original link
IILANE	I	Hold area	Input, original lane
MMLT	I	Hold area	Input, last vehicle in lane of old link
MM	I	Hold area	Input, subject vehicle
L1	I	Switch	Input, code (0,1) if original link is (freeway, ramp)
KXB	I	Dummy argument	
KX	I	Dummy argument	
LEXIT	I	Switch	Output, code (0,1) if vehicle (has not, has) left the free- way

Common Blocks and Variables Referenced

/A1/ VF

/A2/ ENDNDS

/A8/ IPACK, LNMNSP, VFHF

/I10/ LTRNV, MNUVR, RTRNV, THRU

/NFILTS/ NDSCHG

/SIZE/ NVFO

/VBUF/ NF

5.47 Subroutine LSWCH

Program Narrative

Purpose

LSWCH decides whether a given vehicle desires to change lanes and if so, whether it is possible for it to change.

Assumptions

LSWCH assumes that the link and vehicle environment areas contain all information pertaining to the given vehicle.

Approach

First, LSWCH decides whether the vehicle would like to switch to the right, the left, or to either lane. It then checks the desired lanes to see if the given vehicle can make the change. Flags are set to indicate the desirability and possibility of changing lanes.

Synopsis

First, the desirability of changing lanes is examined. A thru vehicle may wish to change if its lane is channelized for turners or is closed, or if it is in the leftmost lane and may be blocked by left-turners downstream. Also, if a thru vehicle is impeded by its leader, it may change lanes. A turner may change if it is not in the extreme left or right lane required for it to turn at the downstream end. Then the possibilities must be checked.

For a thru vehicle, which may switch to the left, LSWCH compares the left leader with the current leader (or, if there is no leader, with a synthetic gap) to determine if the left leader would cause more impedance than the current leader (assuming a left lane exists). Then, the left follower is examined -- it must have an acceptable lag so that the given vehicle can switch lanes. If conditions are favorable, then the vehicle will move to the left. If not, and if the vehicle may switch to the right, or if it may switch only to the right, then the right lane leader and

follower are examined in similar fashion. A flag is set to indicate left, right or no lane switch.

If the vehicle is a left-turner, then the left leader is examined only to see if it is too close to the given vehicle to prevent a safe lane switch (it is not compared with the current leader). Then the left follower is checked in the same way as for thru vehicles.

If the vehicle is a right-turner, then similar checks are made as for left-turners.

Finally, when all flags have been set, the ISLOW flag is cleared if the vehicle was able to change lanes, and control returns to the calling program.

Models and Algorithms

None.

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Lane number of extreme left lane
J	I	Do-loop Index	LLOR index, set in do-loop from 2 to 5
JDIS	I	Hold Area	Gap between vehicle and follower
JTIME	I	Hold Area	Time required to close gap
JVEL	I	Hold Area	Difference between vehicle's and followers' speeds
K	I	Hold Area	LLOR index of vehicle lane

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
XCLEN	I	Hold Area	Vehicle length
LLO	I	Hold Area	Lane number unpacked from LLOR array
LLOCA	I	Hold Area	Position of leader
LLOCI	I	Hold Area	Position of rear of leader
LLOCJ	I	Hold Area	Position of rear of left leader after processing
LLOCJN	I	Hold Area	Position of rear of left leader before processing
LLOCN	I	Hold Area	Position of given vehicle
LPOC	I	Hold Area	Location of beginning of turn pocket

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
ISLOW	I	Switch	Output, (0,1) if file (is not, is) in a lane from which it must switch (because of channeliza- tion, etc) but cannot because of traffic conditions, it must then slow down in hopes of changing during next time step

Common Blocks and Variables Referenced

/A8/ CLEN, IALAG, IPACK

/VENVI/ IDLL, IDRL, JVACL, JVACLL, JVACRL,
JVDHL, JVDHLF, JVDLLL, JVDHRF,
JVDHRL, JVAC, JVFS, JVFSF, JVIDL,
JVIDLF, JVIDLL, JVIDRF, JVIDRL,
JVLEL, JVLELL, JVLRL, JVPRLF, JVQL,
JVQLF, JVQLL, JVQRF, JVQRL, JVSPL,

JVSPLF, JVSPLL, JVSPRF, JVSPRL,
JVVCL, JVVCLL, JVVCRL, LCDF, LCFV,
LCPF

/S2/ IDTS

/LENVI/ LANE, LINK, LLNR, LLOR, LTSURF

/SRFLAG/ LTFLAG

/VBUF/ NF

5.48 Subroutine MOEV

Program Narrative

Purpose

MOEV is the main supervisor routine for the movement of all non-freeway vehicles each time step.

Assumptions

None.

Approach

MOEV loops over all surface links, defines the link environment for each link and then processes all vehicles on the link. The same is done for all ramp links.

Synopsis

First, MOEV loops over all surface links. For each surface link, the link parameter vector is unpacked. The LLOR array, giving the lane numbers in right-to-left order, is defined. SELEN is called to define the link environment. Then MOEV loops over all lanes on the link. The farthest downstream vehicle is found, and MOOV is called to process this vehicle and all successive followers in that lane. The link parameter vector is packed, and the link environment variables are restored by the RELEN routine.

The processing for ramp links is the same except that, after the link parameter vector is unpacked, it must be rearranged to conform to the surface link format. When the data is repacked, these changes must be taken into account.

Models and Algorithms

None.

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	LLOR index, set in do-loop from 1 to M
IRAF	I	Hold Area	Temporary storage for packing link parameters
II	I	Hold Area	Absolute link number
II0	I	Hold Area	Ramp link parameter No. 10, link receiving second movement vehicles
II1	I	Hold Area	Ramp link parameter No. 11, % of volume making second movement
I6	I	Hold Area	Ramp link parameter No. 6, on/off ramp code
I7	I	Hold Area	Ramp link parameter No. 7, code for left or thru first movement
I8	I	Hold Area	Ramp link parameter No. 8, link receiving first movement vehicles
I9	I	Hold Area	Ramp link parameter No. 9, code for thru or right second movement
J	I	Counter	Number of entries in LLOR for thru lanes
K	I	Hold Area	Lane number of left or right pocket
LASTV	I	Hold Area	Argument from LASTV, not used

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
L	I	Hold Area	Link number
LTYP	I	Hold Area	Link type from LASTV, not used
M	I	Hold Area	Number of thru lanes
VL	I	Hold Area	Farthest downstream vehicle
VM	I	Hold Area	Farthest upstream vehicle

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS

/LENVI/ IDITR, IOOF, IRALL, JDNODE, LANE,
LINK, LLOR, LTSURF

/A8/ IPACK

/A2/ IVFEET, NTOTL

/VENVI/ LEDIS

/SRFLAG/LTFLAG

/SIZE/ MXLRA, MXLSU, NLFR, NLRA

5.49 Subroutine MOOV

Program Narrative

Purpose

MOOV simulates the movement of all vehicles in a given lane of a given surface or ramp link for one timestep, including lane changing, discharging from a link, entering and discharging from a queue, and all other intricacies involved in vehicle movement. Service routines are called where necessary to perform many of these functions.

Assumptions

MOOV assumes that all link information necessary to process the vehicle is contained in the link environment common block.

Approach

The unconstrained, free-flow vehicle trajectory is computed. Then it is adjusted according to whether there is a leader, whether the vehicle must slow because of an impending turn, or prohibitive signal. If the vehicle will reach the end of the link, decisions must be made as to whether it will discharge. If so, it is moved along the new link for the time remaining in the timestep. In all cases, the appropriate statistics are updated. If the vehicle is leaving the end of an on-ramp, its entry onto the freeway must be flagged so that the freeway routines will handle it.

Processing begins with the farthest downstream vehicle and continues upstream until all vehicles in the lane have been processed.

Synopsis

Processing begins with the farthest downstream vehicle. The vehicle environment is established by calling INPAK and SEVEN subroutines.

NORM is called to compute the unconstrained, free-flow vehicle trajectory. Then a variety of different paths may be taken to modify this trajectory depending on

whether this is the lead vehicle, whether the end of the link will be reached, what the signal setting is, etc. Many of the paths overlap. This should be kept in mind as the various paths are discussed. Also, areas of common coding are entered and exited from different paths, with switches set to preserve the origin of control.

First, consider the case where there is a leader, ML, in front of the vehicle, M, being processed. The gap between ML and M is computed. If the gap is at least 200 feet, then the normal trajectory is acceptable. If ML is stopped or in queue, then vehicle M must decelerate to avoid collision (error messages are printed in case of collisions and other contradictory situations). If ML is moving and not first in queue, then the trajectory is adjusted to allow for a safe following speed and distance (see 1.5(a) for the car-following rule).

Processing continues now along the same paths for vehicles with and without a leader. TSIG is called to determine attributes of the signal setting, whether it is prohibitive or permissive, whether it just became permissive, and whether it was a yield, stop or right-turn-on-red signal. If the signal just turned prohibitive, but the vehicle is too near the stopline to stop, it is allowed to continue through the intersection. Similarly, it will continue through if it must decelerate at too high a rate in order to stop. Otherwise, if the signal is prohibitive, the vehicle will be slowed to a halt. If the signal is permissive, the vehicle continues through the intersection unless it is first in queue and has not exhausted its lost time, or unless it is a left-turner which will not jump.

If the vehicle is to stop for a prohibitive signal, its stopline target speed is set to zero. If it must slow to negotiate a turn, then the target speed is set appropriately. The final speed is adjusted as necessary to achieve this target speed. (See 1.5 (b) for algorithm.)

At this point, processing can again follow many paths. If the vehicle is not first in queue and will not reach the end of the link this timestep, then LSWCH is called to determine whether the vehicle will change lanes. If it is to change to the left, then it is flagged as having done so, but it will actually be processed when MOOV is called for

that lane. Hence, processing for this vehicle is finished for now. If it is changing to the right, then the environment must be adjusted accordingly, and processing is restarted as described above for the vehicle in its new lane. Finally, if no lane change occurred, additional adjustments may be made to the trajectory depending on conditions in the next link. Link statistics are updated, and processing is finished for this vehicle.

Now, suppose the vehicle will reach the end of the link. If it is first in queue and the headway has not expired or if the previous leader did not discharge, then it will not discharge. If it is not in queue, then a moving discharge is indicated. No vehicle can discharge, of course, if the signal is prohibitive. If it is permissive, TSTSAT is called to determine if there is space in the next link. If TSTSAT permits discharge and the vehicle is discharging onto a freeway, it must be flagged in the IORG array to be processed later by the freeway routines, and processing here is over except for updating of statistics. If the vehicle is to remain a non-freeway vehicle, then GOQ must also be called to check for acceptable gaps and cross traffic blockage. If any of these tests fail, the vehicle will not discharge, so statistics are updated and processing is over for the vehicle.

All possibilities for the vehicle on its original link have now been covered. Suppose now that the vehicle will discharge. All link statistics relating to the original link are updated, and the time remaining for processing on the new link is computed. If the vehicle is exiting from the system, processing is essentially finished; if not, it must be moved along its new link. First, LANES must be called to determine the lane assignment on the new link. NORM is called to determine the normal, free-flow trajectory on the new link. Then, with flags set to indicate that the vehicle is on the new link, the same code as before is executed to adjust the trajectory if there is a leader.

If the vehicle which has just discharged has a follower, then HDWY is called to determine the discharge headway for that vehicle.

Finally, the vehicle and link environments are updated to reflect the vehicle's move. If the vehicle has

changed from one link type to another, then a new vehicle number must be found. The vehicle data is packed, and processing continues in the same fashion for each successive follower, until all vehicles in the given lane have been processed.

Models and Algorithms

(a) The car-following law requires a vehicle speed and gap between leader and vehicle such that, if the leader should decelerate to a halt at 14 fpss, the given vehicle, after completing its movement this timestep, could decelerate at 7 fpss and still avoid collision with the leader. The required speed is found by solving simultaneous equations:

$$P_2 - P_1 = \frac{v_1^2}{14}$$

where P_2 is leader position after emergency deceleration,

P_1 is vehicle position after normal movement

v_1 is required vehicle speed

$$P_1 - P_0 = \frac{(v_1 + v_0)}{2} t$$

where P_0 is initial position of vehicle,

v_0 is initial speed of vehicle,

t is time interval

Combining the two, we get

$$P_2 - P_1 = \frac{v_1^2}{14} + v_1 \frac{t}{2} + v_0 \frac{t}{2}$$

and we solve the quadratic to get v_1 .

(b) Similarly, to find the speed required at the end of the timestep such that the vehicle can reach its target speed at the stopline, we must solve the following for v_1 :

$$P_2 - P_1 = \frac{v_1^2}{14} - \frac{I^2}{14}$$

where

P_2 is the stopline location
 P_1 is vehicle position after movement
 I is the target speed at stopline

$$P_1 - P_0 = \frac{(v_1 + v_0)}{2} t$$

where

P_0 is initial vehicle position
 V_0 is initial vehicle speed
 t is time interval

Combining, we get

$$P_2 - P_0 = \frac{v_1^2}{14} - \frac{I^2}{14} + \frac{v_1 t}{2} + \frac{v_0 t}{2}$$

$$\text{or } 14(P_2 - P_0) = v_1^2 - I^2 + 14 \cdot \frac{v_1 t}{2} + 14 \cdot \frac{v_0 t}{2}$$

We solve the quadratic to get the required speed v_1 .

Date, programmer

January, 1976. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Constant	1/14, set in data statement, used in quadratics
AA	R	Hold area	AA, B, C are computed inputs to QUAD function
B	R	Hold Area	
C	R	Hold Area	
DD	R	Hold Area	Vehicle length
IDD	I	Hold Area	Negative of distance to downstream end of link
IDIS	I	Hold Area	Distance travelled in one time step
IDLENG	I	Hold area	Vehicle type code
IDSD	I	Hold area	Stopping distance for deceleration at 7 fpss
IDSL	I	Hold Area	Distance to stopline
IDSTIM	I	Hold Area	Minimum discharge headway
IDTQ	I	Hold Area	Portion of time step for which vehicle is being moved
IHAF	I	Hold area	Temporary storage of parameters to be packed
IHY	I	Hold Area	Discharge headway

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IMGCY	I	Hold Area	Emergency deceleration
IPATH	I	Switch	(1,2) if processing portion of vehicle's path on (original, new) link
IQ	I	Do-loop index	Lane position index, set in do-loop from 1 to 5
IQL	I	Hold Area	Lane index of desired lane
IQMX	I	Hold Area	Lane index of left-most lane
ISLOW	I	Switch	(0,1) if vehicle (is not, is) in a lane from which it is prohibited and must slow down to change lanes as soon as possible
ITRNSP	I	Switch	(1,2) if vehicle is slowing to (stop, turn right or left)
ITTS	I	Hold Area	Time required to stop
IVFLAG	I	Switch	(1,2) if first calculation of final speed (was not, was) changed for first vehicle in queue
J	I	General-purpose hold Area	
JDTQ	I	Hold Area	One-half of time step, used for rounding computed acceleration
JHAF	I	Hold Area	Grade code index

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JJ	I	Hold area	Downstream lane number; also random number
JJJ	I	General- purpose Hold Area	
JVFLNO	I	Hold Area	Final lane on original link
K	I	General- purpose Hold Area	
KDIS	I	Hold Area	Distance from stopline
KDIST	I	Switch	(1,2) for (discharge from queue, moving discharge)
KK	I	Hold Area	Final acceleration, output from NORM
KSIGC	I	Hold Area	Original value of JSIGC, output from TSIG routine
KVF	I	Hold Area	Leader-based speed or target speed
LNNK	I	Hold Area	Absolute link number
M	I	Hold Area	Vehicle Number
MCS	I	Hold Area	Vehicle initial speed
MDS	I	Hold Area	Desired free-flow speed
MFLAG	I	Switch	(1,2) if (leader in motion, slowing down to turn)
MGAP	I	Hold Area	Distance between vehicle and leader
MGC	I	Hold Area	Grade code index

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ML	I	Hold Area	Original vehicle number to be processed
MNEW	I	Hold Area	Vehicle number on new link
MPA	I	Hold Area	Present acceleration of vehicle
MVT	I	Hold Area	Vehicle type code
NWLNK	I	Switch	(1,2,3) if vehicle will move onto (left, thru, right) receiving link
VO	R	Hold Area	Initial speed
X	R	Hold area	Discriminant of quadratic solution
Y	R	Hold Area	Numerator of quadratic solution
Z	R	Hold Area	Quadratic solution, time required for leader to clear stop line (seconds)

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
MM	I	Hold Area	Input, surface or ramp vehicle to be processed

Common Blocks and Variables Referenced

/A8/CLEN,ILT,IPACK,MINAC,TLNK,VFH
 /LENVI/IDITR,IOOF,LACR,LANE,LDTR,LEXA,
 LFFR,LFTR,LINK,LLNA,LLOR,LMTR,LPOR,
 LPRR,LSPR,LUVR,LVLR

/VENVI/IDLL, IDRL, JPFOL, JPLED, JSIGC, JSIGP,
JVDHF, JVDHL, JVFAC, JVFFO, JVFLE, JVFLK,
JVFLN, JVFLT, JVFLTY, JVFOL, JVFOLL, JVFORL,
JVES, JVFSP, JVIDF, JVIDL, JVIDLF, JVIDLL,
JVIDM, JVIDRF, JVIDRL, JVILT, JVLC, JVLEF,
JVLEL, JVLELF, JVLRF, JVQF, JVQL, JVSPF,
JVSPL, JVVCL, LCDF, LCFV, LCPF, LEDIS, NEVE

/S2/IDTS, ITIME, ITIMEF

/ERTRAN/IEPAR, IER, IERCT, NPAR

/ONVEH/IORG, IREN, IRV, NORG

/LASDCH/LADISH

/SRFLAG/LTFLAG

/NFILTS/NDSCHG

/VBUF/NF, NRL

/SIZE/NLFR, NLRA

5.50 Subroutine NORM

Program Narrative

Purpose

NORM computes the normal, free-flow trajectory for a given vehicle.

Assumptions

NORM assumes that all arguments passed to it through the calling sequence are valid.

Approach

NORM computes new speed, acceleration, and distance travelled to approach the desired speed, within limits required by certain calibrated parameters.

Synopsis

If the given time step is zero, then the vehicle cannot move at all, so the new speed and acceleration are set equal to the current values and processing is over. Otherwise, the acceleration or deceleration required to attain the desired speed is computed. If the vehicle must decelerate, it must not decelerate at more than the allowed coasting deceleration rate. If it will accelerate, it is limited by its maximum acceleration and jerk. Then the new speed and distance travelled are computed in the usual manner and returned through the calling sequence.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
II	I	Hold area	Round-off factor for time step, equal to IIT/2
ISP	I	Hold area	MAXAC speed index

Arguments passed to subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IA	I	Hold area	Output, new acceleration
ICS	I	Hold area	Input, current speed
IDIS	I	Hold area	Output, distance travelled
IDS	I	Hold area	Input, desired speed
IGCT	I	Hold area	Input, grade code
INF	I	Switch	Input, (0,1) for (freeway, non-freeway) trajectory
INS	I	Hold area	Output, new speed
IPA	I	Hold area	Input, present acceleration
IT	I	Hold area	Input, time step
IVT	I	Hold area	Input, vehicle type

Common Blocks and Variables Referenced

/A8/CSTDCL, MAXAC

5.51 Subroutine OFFRMP

Program Narrative

Purpose

OFFRMP is called to place vehicles exiting the freeway on the off-ramp and to update relevant statistics.

Assumptions

It is assumed that the IPACK and KROSBF arrays have been primed with the parameter vectors for the freeway and ramp links, respectively. Also, OFFRMP assumes that the NNF array contains the existing ramp vehicle parameters.

Approach

OFFRMP locates the leader vehicle on the ramp and adjusts the subject vehicle's deceleration so as to achieve a safe separation and speed.

Synopsis

First, the ramp vehicle which is last in lane is identified and its parameters are unpacked. The leader-follower parameters for this vehicle and the subject vehicle are set. A target stopping point is calculated corresponding to the position of the leader's tail after an emergency stop. In the absence of a leader, the downstream end of the off-ramp is adopted as the target stopping point.

The speed of the subject vehicle, as it entered the offramp, is then used to determine if it could stop before reaching the target stopping point. If not, the acceleration during the current time-step is adjusted to provide a safe trajectory.

If any trajectory adjustment is required, the vehicle-minutes statistics for the original freeway link are revised.

All link statistics for the ramp are updated. Link dependent vehicle parameters (i.e., desired speed, turn movement) are generated for the subject vehicle.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	General Purpose Hold Area	
AA	R	General Purpose Hold Area	
B	R	General Purpose Hold Area	
C	R	General Purpose Hold Area	
I	I	Hold Area	Acceleration or deceleration of vehicle during this time-step, fpss
IAS	I	Hold Area	Starting position of vehicle, feet
IAV	I	Hold Area	Starting speed of vehicle, feet
IHAF	I	Hold Area	Number of new ramp vehicle
ISBF	I	Hold Area	New vehicle speed at boundary, fps
ISBI	I	Hold Area	Old vehicle speed at boundary, fps

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ITARG	I	Hold Area	Target stopping point of vehicle, M, feet
ITF	I	Hold Area	New time on old link, tenths-of-a-second
ITI	I	Hold Area	Old time on old link, tenths-of-a-second
J	I	Hold Area	Actual stopped position of vehicle, feet
K	I	Switch	From Code (0,1) if (first, second) movement
LO	I	Hold Area	Old freeway link number
LRV (25)	I	Hold Area	Array containing unpacked lead vehicle parameters
M	I	Hold Area	Number of new ramp vehicle
MDEC	I	Hold Area	Maximum deceleration for lead vehicle, ML, fpss
MDT	I	Hold Area	Driver type code of vehicle, M
ML	I	Hold Area	Lead vehicle to vehicle, M
MPOS	I	Hold Area	Distance of vehicle, ML, from upstream node, feet
MTT	I	Hold Area	Total moving time for vehicle, M, tenths-of-a-second
MVT	I	Hold Area	Vehicle type code of vehicle, ML
SA	R	Hold Area	Starting position of vehicle, feet

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
T	R	Hold Area	Freeway time step, seconds
TAR	R	Hold Area	Target stopping point of vehicle, M, feet
VA	R	Hold Area	Starting speed of vehicle, fps

Arguments Passed to Subroutines

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KROSBF (47)	I	Hold Area	Input, array containing freeway link parameters for link LOLD
LOLD	I	Hold Area	Input, freeway link number upstream of ramp
MM	I	Hold Area	Input, number of new ramp vehicle
NNF (25)	I	Hold Area	Input, array containing unpacked ramp vehicle parameters for vehicle, MM

Common Blocks and Variables Referenced

/A2/ IVFEET

/A8/ CLEN, IPACK, MINAC, VFH

/ERTRAN/ IEPAR, IER, NPAR

/S2/ IDTF, ITIME, ITIMEF

/SIZE/ NLFR

5.52 Subroutine ONRMP

Program Narrative

Purpose

ONRMP emits onto the freeway all vehicles which have been discharged from on-ramps and are destined for the specified freeway link lane.

Assumptions

ONRMP assumes that the freeway link parameters have been stored in the IPACK array. Also, it is assumed that the ONVEH COMMON Block has been properly primed with the identities, time-of-discharge, and destination link-lane for all vehicles leaving on-ramps.

Approach

This routine first determines if any vehicles exist which have been discharged from on-ramps bound for the specified freeway link and lane. Upon locating such vehicles, ONRMP locates them on the freeway link, packs the new freeway vehicle parameter vector and deletes the ramp vehicle-parameter vector.

Synopsis

ONRMP first tests whether this freeway link and lane is serviced by an on-ramp. If not, it returns to CALLING routine. Then the vehicle furthest upstream on the specified lane of the freeway link is located. The vehicle farthest downstream on the feeder link and lane which feeds the subject lane is then identified.

ONRMP then loops over the buffer region containing the identities of vehicles which have been discharged from on-ramps and are now awaiting merge entry into freeway links. Those vehicles, if any, which are available to enter this freeway link and lane are identified. For each such vehicle the parameter vector is unpacked using routine UNPAK

Each vehicle is located on its receiving freeway link so that its position relative to its leader is sensible (i.e., safe from an emergency deceleration standpoint). This vehicle's position is then checked with that of the following vehicle, and adjusted if needed to assure that any turbulence will be minimized.

Routine FINDSV is then called to locate an empty slot for this vehicle in the freeway vehicle array. The follower and leader vehicles are then located and their parameters adjusted to establish the chain of vehicles. The follower vehicle on the ramp, if any, is located. If that vehicle is in queue, its discharge headway is calculated and its statistics are adjusted accordingly. If the ramp vehicle is still moving and not in queue, its statistics are adjusted to reflect the absence of a leader. The freeway link statistics are adjusted to reflect the entry of the subject vehicle.

Appropriate input parameters are primed and then routine COLECT is called to determine whether a detector has been actuated by this new freeway vehicle. The subject vehicle is then removed from the buffer array and from the ramp vehicle array.

Models and Algorithms

None.

Date, Programmer

June 1976. David Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Dummy Argument to RANDOM	
IA	I	Hold Area	Acceleration or deceleration of vehicle during this time-step, fpss
ICS	I	Hold Area	Current vehicle speed, fps
IDIS	I	Hold Area	Distance travelled, feet
IDS	I	Hold Area	Desired free-flow speed, fps
IGCT	I	Hold Area	Grade code
II	I	Hold Area	Vehicle type code
IL	I	Hold Area	Current freeway link
ILKF	I	Hold Area	Current freeway link
ILKL	I	Hold Area	Length of freeway link, feet
INS	I	Hold Area	New speed, fps
IPA	I	Hold Area	Present acceleration, fpss
IPOS	I	Hold Area	Position of follower relative to upstream node of link LK.
IPRF	I	Switch	Process Code (0,1) if vehicle (has not, has) been processed this time-step

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IPSF	I	Hold Area	Distance of vehicle from upstream node, feet
ISEP	I	Hold Area	Resulting separation, feet
ISF	I	Hold Area	Current speed of follower, fps
IT	I	Hold Area	Time-step, tenth of a second
IVT	I	Hold Area	Vehicle type code
J	I	Do Loop Index	General purpose do-loop index
JJ	I	Hold Area	Vehicle type code
JPOS	I	Hold Area	Projected position of follower
K	I	Hold Area	Driver type code
KRUB	I	Dummy Argument to BLOK	
L	I	Hold Area	Number of lanes
LASL	I	Dummy argument to FRSTV, LASTLK, LASTV	
LEDLAN	I	Hold Area	Current lane
LINDEX	I	Hold Area	Number of vehicles in front of this vehicle in current lane
LK	I	Hold Area	Freeway link being entered
LKTYP	I	Switch	(0,1) if feeder lane is (freeway, ramp)
LMNSPD	I	Hold Area	Lane mean speed, fps

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LN	I	Hold Area	Lane number
LPREV	I	Hold Area	Upstream lane number feeding LN
LRLEN	I	Hold Area	Length of ramp, feet
LRVPO	I	Hold Area	Distance of vehicle from upstream node, feet
LRVSP	I	Hold Area	Current speed, fps
LTYPE	I	Dummy Argument to LASTV	
LUPLAN	I	Hold Area	Upstream lane number feeding LN
M	I	Hold Area	Number of vehicles in front of this vehicle in current lane
MF	I	Hold Area	Number of vehicle behind this vehicle in current lane
MM	I	Hold Area	Farthest upstream vehicle
MR	I	Hold Area	Ramp vehicle number of vehicle entering freeway link
NELAN	I	Hold Area	Thru receiving lane
NELIN	I	Hold Area	Thru receiving link
NETYP	I	Hold Area	Receiving link type, (0,1,2) for (freeway, ramp, exit on non existent) link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ILK	I	Hold Area	Input, freeway link number
ILN	I	Hold Area	Input, lane number

Common Blocks and Variables REferenced

/A1/ LNKF,VF

/A2/ ENDNDS,IVFEET,

/A3/ SIGI

/A8/ CLEN,IPACK,LNMNSP,VFHF

/A10/ MNUVR

/FPAR/ NCH

/FORINC/ ICUR,IPOS1,IPOS2,ISPD1,ISPD2,
ISTART,ISTEP,IVNUM,LANE1,LANE2,
LENVH,LINK1,LINK2,NCHNG

/ONVEH/ IORG,IREN,IRV,NORG

/S2/ IDTF,ITIME,ITIMEF

/SIZE/ NLFO,NVFO

/VBUF/ NF,NFA,NRL

5.53 Subroutine PDAFZ

Program Narrative

Purpose

To calculate the minimum inter-arrival gap for all detectors that are servicing the current active phase.

Assumptions

Assumes DETECT routine has provided pertinent detector information.

Approach

This routine scans through all detectors servicing a current active phase and calculates the minimum actuation time among all those detectors. This routine calls FZCL to determine whether there is a need to start the maximum green timer.

Synopsis

The routine scans all detectors servicing the active phase, I2, at the actuated node, I1. For each detector, current time actuation is calculated. When all detectors are processed, a minimum actuation time is retained and stored. Routine FZCL is called to determine whether any other phase has been called if the maximum green timer for phase I2 has not yet started. Start the maximum green timer for phase, I2, if there indeed has been an opposing call.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General Purpose Hold Area and Do-loop Index	
J	I	Hold Area	Packed word containing speed and time of actuation
JDEL	I	Hold Area	Inter-arrival gap
JJ	I	Hold Area	Unpacked overlapping code
JDMIN	I	Switch	Memory switch, code (999,3) if (on, off)

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, phase identification
I3	I	Hold Area	Input, status code

Common Blocks and Variables Referenced

/ACT1/ NAFZ
/ACT3/ NATMR
/ACT4/ NADET
/ACT5/ NADTM
/ACT15/ NEWDET
/S2/ ITIME

5.54 Subroutine PDNAFZ

Program Narrative

Purpose

To examine all detectors servicing each inactive phase and if an actuation took place within the past time-step, PDNAFZ will update the appropriate counters.

Assumptions

None.

Approach

This routine examines all detectors servicing each inactive phase. The actuation counters for each detector are incremented if an actuation took place within the past time-step.

Synopsis

The routine examines all detectors. If it is a counter detector, the detector actuation counter will be incremented only if an actuation took place during the last step. If it is a presence detector, the detector actuation counter will contain a code of 1 as long as there still are vehicle(s) in the presence detector trap.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Detector index, set in do-loop from 1 to 8
J	I	Hold Area	Time of Actuation

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, current inactive phase number on actuated node I1

Common Blocks and Variables Referenced

/ACT4/ NADET

/ACT5/ NADTM

/ACT6/ NACNT

/ACT15/ NEWDET

5.55 Subroutine QSTATE

Program Narrative

Purpose

QSTATE determines whether or not a vehicle is about to join a queue.

Assumptions

QSTATE assumes that the link and vehicle environment areas contain all information pertaining to the given vehicle.

Approach

QSTATE examines the vehicle's current state and its relation to its leader to decide whether it will join a queue.

Synopsis

If the given vehicle is already in queue, no further checking is needed -- it remains in queue. If the vehicle is travelling at a minimum speed of 10 fps, it is not in queue. If the vehicle's leader is not in queue, or if the leader is 10 feet beyond the given vehicle, the given vehicle is not in queue; otherwise, it is in queue. If there is no leader, however, different tests are made: if the vehicle is accelerating or if it is not within 10 feet of the stopline, it is not in queue. Otherwise, it becomes the first vehicle in queue, and the queue code is set accordingly.

Models and Algorithms

None

Date, programmer

January, 1976. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold area	Leader position
K	I	Switch	(0,1) for vehicle (not in, in) queue

Arguments passed to subroutine

None

Common blocks and variables referenced

/A8/CLEN,IPACK

/VENVI/JVDHL,JVFAC,JVFLE,JVFLN,JVFS,
JVFSP, JVLEL, JVQL, JVSPL, JVVCL,

/LENVI/LANE, LINK

/VBUF/NF

5.56 Subroutine REDSIG

Program Narrative

Purpose

To update the amber, the red clearance and the red revert of an active phase.

Assumptions

Assumes DETECT routine has provided pertinent information regarding detectors.

Approach

This routine updates the amber, the red clearance and the red revert interval for an active phase. The interval is terminated when the specified duration has been attained and the status code is adjusted to reflect the beginning of the next interval.

Synopsis

The routine first retrieves the active phase, NP, and its current interval timer. Then the routine branches according to the status code of amber, red clearance and red revert. The routine ASIG is called to change the link-specific signal from green to red directly when the interval timer has just crossed the half-way point of the amber duration. This is done because there is no movement-specific amber signal code. In the case of a red clearance interval without a red revert interval, the routine will set the absolute termination code when the red clearance interval has been completed. However, in the case of a red clearance with a red revert interval, the processing is slightly different. After completion of the red clearance interval, the absolute termination code will be set only if either there is an outstanding phase call which is indicated by FZCL routine or the revert interval has been completed.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JK	I	Hold Area	Status code
JM	I	Hold Area	Amber duration
K	I	Hold Area	Operation code
LL	I	Hold Area	Active phase
MTMR	I	Hold Area	Timer for active indication

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, ring identifier
I3	I	Hold Area	Input, status code

Common Blocks and Variables Referenced

/ACT1/ NAFZ
/ACT2/ NAAFZ
/ACT3/ NATMR
/ACT12/ NASTAT
/ACT16/ MTERM

5.57 Subroutine RELEN

Program Narrative

Purpose

RELEN repacks the link environment parameters into the link array.

Assumptions

RELEN assumes that the link environment contains the latest link parameters.

Approach

RELEN loops over all turn movements and packs the parameters for each into the appropriate array.

Synopsis

RELEN loops over all turn movements (left, thru, and right). For each movement, it calls the appropriate packing routines, depending on whether the link is a surface or ramp link, to repack all the updated parameters.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	Do-loop index	Turn movement index, set equal to (1,2,3) for (left, thru, right)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KM	I	Hold Area	Parameter number of link receiving turn movement
L	I	Do-loop index	Lane number, set in do- loop from 1 to 5
LK	I	Hold Area	Link number receiving turn movement

Arguments passed to subroutine

None

Common Blocks and Variables Referenced

/LENVI/IOOF,IRALL,JDNODE,LDHR,LDTR,LFTR,
LINK,LMHR,LMTR,LSHR,LSTR,LTSURF,LUVR

/A8/IPACK

/A2/IVFEET

/SIZE/NLFR,NLRA

5.58 Subroutine REVEN

Program Narrative

Purpose

REVEN repacks the updated vehicle environment parameters.

Assumptions

REVEN assumes that the environment contains the latest values of all leader and follower parameters.

Approach

REVEN processes the data according to the vehicle type.

Synopsis

REVEN performs the same functions for ramp and surface vehicles, except that different packing routines must be called. REVEN loops over the left, current and right lanes and packs the required follower and leader parameters.

Models and Algorithms

None.

Date, Programmer

January, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
K	I	Do-Loop Index	Lane index, set in do-loop equal to (1,2,3) for (left, current, right) lane
ITEMP2	I	Hold Area	Temporary storage used for re-packing parameters

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/VENVI/ JPFOL, JPLED

/SRFLAG/ LTFLAG

5.59 Subroutine RISK

Program Narrative

Purpose

RISK determines the maximum acceptable deceleration for the subject vehicle or its new follower in the target lane when a lane change is being tested.

Assumptions

RISK assumes all arguments passed to it are valid.

Approach

The deceleration of a vehicle is obtained based on its maximum allowable deceleration and the required deceleration to avoid collision.

Synopsis

If the vehicle is the lane changer, its acceptable deceleration is determined on the basis of whether the lane change is mandatory or not. The driver type is also a factor. If the vehicle is a follower to the lane changer in the target lane, it is determined whether the required acceleration is acceptable by calling routine CHOOZ. CHOOZ determines the cooperative nature of the follower and its willingness to decelerate at a higher rate.

Models and Algorithms

None.

Date, Programmer

January 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDES	I	Switch	(0,1) for driver (not willing, willing to yield right-of-way
JT	I	Switch	(0,1) for (changer, follower) vehicle
KEACC	I	Hold Area	Minimum deceleration (-) in fpss $\times$ 10
KOFF	I	Switch	(0,1) indicates that changer (must, may) change lanes
LD	I	Switch	Driver type code
LEX	I	Hold Area	Location of changer on link
LNG	I	Hold Area	Link length
LV	I	Switch	Vehicle type code
MINF	I	Hold Area	Minimum acceptable accelera- tion (+ or -) in fpss $\times$ 10
MINK	I	Hold Area	Required deceleration of ve- hicle in fpss $\times$ 10

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JJT	I	Switch	Input, (0,1) for (changer, follower) vehicle
KKOFF	I	Switch	Input, (0,1) indicates that changer (may, must) change lanes
KKV	I	Dummy Argu- ment	Input, current speed of vehicle in fps

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LLD	I	Switch	Input, driver type code
LLEX	I	Hold Area	Input, location of changer on link
LLNG	I	Hold Area	Input, link length
LLV	I	Switch	Input, vehicle type
MMINF	I	Hold Area	Output, maximum acceptable deceleration in fpss $\times 10$

Common Blocks and Variables Referenced

/A8/ MINAC

/FPAR/ ISENS, MNAC

5.60 Subroutine SELEN

Program Narrative

Purpose

SELEN determines the approaches to the downstream node of a given link and primes the link environment with information on approach and receiving links.

Assumptions

SELEN assumes that the IPACK array contains the link parameter vector for the given link.

Approach

All the approaches are found and compared with receiving links and turn movements to determine the geometrical configurations.

Synopsis

All the approach links and link types for the downstream node of the given link are found in the SIGI array (the given link itself is eliminated from consideration.). Next, the downstream nodes of the receiving links are found as well as the upstream nodes of the approach links. SELEN can then compare these two sets of nodes to determine, as far as possible, which approach links are the left, thru or right approaches. If any approaches remain whose geometrics have not been determined, then, for each one, the thru receiving link of that approach is compared to the receiving link of the given link; if a match is found, then the geometry is defined (e.g. an approach whose thru link is the left receiving link of the given link is the right approach to that link). Then the same can be done with the right and left receiving links of the approach link.

When all approach and receiving links have been determined, FILL is called for each one to fill the appropriate elements of the link environment.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews

Program Variables

Local variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	Do-loop index	Approach index, set in do-loop from 1 to 4
KK	I	Do-loop index	Approach index, set in do-loop from 1 to 3
KM	I	Hold area	IPACK index of left, thru or right receiving link
L	I	Hold area	Absolute link number
LA(3)	I	Hold area	LA(K) is (left, thru, right) approach link for K = (1,2,3)
LAT(3)	I	Switch	LAT(K) is link type of LA(K)
LATEMP (4)	I	Hold area	Temporary storage of approach links to downstream node
LATYP(4)	I	Switch	LATYP(k) is link type for LATEMP(K)
LC	I	Hold area	Temporary storage of unpacked link parameters
LK	I	Hold area	Receiving link number
LTEM	I	Hold area	Receiving link type

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ND(3)	I	Hold area	Downstream nodes of receiving links
NU(4)	I	Hold area	Upstream nodes of approach links

Arguments passed to subroutine

None

Common blocks and variables referenced

/A2/ENDNDS

/LENVI/IOOF,JDNODE,LADICH,LEXA,LLNR,
LTSURF,LINK

/ERTRAN/IEPAR,IER,NPAR

/A8/IPACK

/SRFLAG/LTFLAG

/SIZE/NLFR,NLRA

/A3/SIGI

5.61 Subroutine SEVEN

Program Narrative

Purpose

SEVEN determines the left and right leaders and followers of the given vehicle and primes the vehicle environment parameters accordingly.

Assumptions

None

Approach

The left and right lanes are determined, the leaders and followers in those lanes are found, and the environment parameters are primed.

Synopsis

First, SEVEN determines the lanes to the immediate right and left of the given vehicle. Then, for each lane, SEVEN examines the farthest upstream vehicle and each successive vehicle until the first one is found whose position is farther downstream than the given vehicle; this is the left or right leader, and its follower is the left or right follower. Now SEVEN primes all environment parameters for the left, current and right leaders and followers with values unpacked from the appropriate vehicle array.

The given vehicle's final speed, position, and other parameters are primed with the current values; they will be updated by MOOV.

Models and Algorithms

None

Date, programmer

January, 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ICODE2	I	Switch	(0,1) if vehicle (is not, is) in queue
K	I	Do-loop index	Index for left, current or right leader or follower
KAUX	I	Hold Area	Argument to LASTV, not used
L	I	Do-loop index	Leader or follower parameter index, set in do-loop from 1 to 8
LDIS2	I	Hold Area	Distance or headway
LLO	I	Hold Area	Lane number from LLOR array
LNUM(2)	I	Hold Area	LNUM(K) is (left,right) lane for K = (1,2)
LPOS	I	Hold Area	Position of given vehicle
MF	I	Hold Area	Left or right follower of given vehicle
ML	I	Hold Area	Left or right leader of given vehicle
MLAST	I	Hold Area	Last vehicle examined
ML2	I	Hold Area	Leader of vehicle being examined
MSP2	I	Hold Area	Speed of vehicle being examined

Arguments passed to Subroutine

None

Common Blocks and Variables Referenced

/VENVI/IDLL, IDRL, JPFOL, JPLED, JVFAC, JVFFO,
JVFE, JVFLK, JVFLN, JVFLT, JVFLTY, JVFS,
JVFSP, JVIDF, JVIDL, JVIDLF, JVIDLL, JVIDRL,
JVILT

/S2/IDTS

/ERTRAN/IEPAR, IER, NPAR

/A8/IPACK

/LENVI/LINK, LLOR

/SRFLAG/LTFLAG

/VBUF/NF, NFA

5.62 Subroutine SIGACT

Program Narrative

Purpose

To control the processing of an actuated controller.

Assumptions

SIGACT assumes that there is an actuated node.

Approach

This routine is entered to initiate processing of the specified actuated controller. Its primary function is to scan each phase of the controller. If a phase is non-actuated, it is not processed in this routine. If it is actuated and active, PDAFZ is called to update all detectors that are servicing this active phase. If it is actuated and not active, PDNAFZ is called to update all the actuation counts for those detectors that are servicing this inactive phase. After all the phases have been examined, DETSW is called if the detector-switching option is active. Routine UPACT is then called to update the current status of the controller.

Synopsis

The routine loops through each actuated phase and examines its status code. If the status code indicates that the phase in question is active, PDAFZ routine is called to update all detectors that are servicing this phase. Otherwise, the phase is inactive and PDNAFZ routine is called to update all actuation counts for those detectors that are servicing this inactive phase.

When all actuated phases are examined, including those phases on second ring if it is a dual ring controller, the DETSW routine is called, if the detector switching option is selected by the user.

Finally, the UPACT routine is called to update the status for the active phase.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IPH	I	Do-loop Index	Phase number, set in loop from 1 to 4
IR	I	Do-loop Index	Ring number, set in loop from 1 to IRNGS
IRNGS	I	Hold Area	Number of rings
K	I	Hold Area	Status code
N	I	Hold Area	Actuated controller number
NP	I	Counter	Phase counter

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, node number where actuated controller is located

Common Blocks and Variables Referenced

/ACT1/ NAFZ
/ACT2/ NAAFZ
/ACT3/ NATMR
/ACT8/ NACDS
/ACT10/ NACT
/ACT12/ NASTAT
/ACT18/ ND

5.63 Subroutine SVEH

Program Narrative

Purpose

SVEH generates new vehicles on surface entry links.

Assumptions

None

Approach

SVEH loops over all surface entry links and, if it is time, generates new vehicles and stores their information in the surface vehicle array.

Synopsis

SVEH loops over all entry links, skipping those that are not surface entries. Each surface entry is processed as follows: the time remaining till next emission is decremented, and if the time has not expired, it is stored and processing is over for this entry link. If it has expired, then the link data is unpacked. FINDSV is called to find an empty slot in the surface vehicle array. The vehicle parameters are initialized, to be later packed. The driver type, vehicle type, free-flow speed, and turn movement are all based on uniform random number distributions. If there is a leader vehicle in the entry link, then the new vehicle position must take the leader into consideration. (If there is no room for the new vehicle in any lane, the simulation is aborted.) Leader and follower parameters are set appropriately. The MNUVR is updated to reflect the new vehicle's turn movement. The vehicle data is packed, and the new vehicle becomes the farthest upstream vehicle in its lane.

The time until next emission is incremented by the emission headway. If time allows, then another vehicle is generated on the entry link. If not, the time is

stored and processing goes on to the next link.

Models and Algorithms

None

Date, programmer

January 1976. Barbara Andrews

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Time remaining till next emission
IPKT	I	Hold Area	Turn pocket capacity
IR	I	Hold Area	Temporary storage for unpacking vehicle type percentages
ISUM	I	Hold Area	Sum of vehicle type percentages
IV	I	Hold Area	Leader vehicle number
IVTPC (5)	I	Hold Area	Percentages for each of five vehicle types
I2	I	Hold Area	Leader of upstream vehicle
J	I	Do-loop index	Entry link index, set in do-loop from 1 to NTOTN
JJ	I	Do-loop index	Vehicle type percentage index

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	General- purpose hold area	
KK	I	Hold Area	IS array index
KM	I	Hold Area	Index to farthest upstream vehicle
L	I	Hold Area	Link number
M	I	Do-loop index	Lane index, set in do-loop from 1 to MM
MM	I	Hold Area	Number of lanes in link
MR	I	Hold Area	Uniform random number between 1 and 1000 (not used)
MV	I	Hold Area	Farthest upstream vehicle in lane
NEWV	I	Hold Area	Vehicle number of newly- generated vehicle same as NEWV
NR	I	Hold Area	Uniform random number between 1 and 100
RNU	R		Uniform random number between 1 and 100 (not used)

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/A8/CLEN,IPACK,VFH

/A2/ENDNDS

/S2/IDT, IDTS, ITIME

/ERTRAN/IEPAR, IER, NPAR

/SIZE/IS

/A10/MNUVR

/VBUF/NF, NRA, NRL

/SIZE/NLFR, NLRA

/A7/NS, SNODE

/A3/NTOTN

5.64 Subroutine TERM

Program Narrative

Purpose

To either terminate a currently active phase or move a phase from green to amber interval.

Assumptions

TERM assumes that either the current phase is active or the phase is green.

Approach

This routine changes the status code of a given phase to either the red (inactive) or the amber interval. It also clears the relevant timers and the actuation counters.

Synopsis

The routine changes the status code for the phase, I2, according to the code, I3, and it clears all the timers except the cycle timer. If I3 is positive, the routine then removes the phase, I2, from the active state. If I3 is negative, the routine clears all the actuation counters before returning to the calling routine; the phase remains active.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold Area	Status code = (7,4) if phase is (inactive, in clearance)
IR	I	Hold Area	Absolute value of code I3
K	I	Hold Area	New status code for NP

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number
I2	I	Hold Area	Input, phase identifier
I3	I	Hold Area	Input, status code

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT6/ NACNT

/ACT12/ NASTAT

5.65 Subroutine TERMFZ

Program Narrative

Purpose

To terminate those phases that were flagged with the absolute termination code and to activate new phases.

Assumptions

Assume DETECT routine has provided pertinent information about the detector.

Approach

In the case of a single ring controller, the routine terminates the phase that was flagged with the termination code and activates the next phase that was called. In the case of a dual ring controller, the routine differentiates whether one phase or two phases need to be terminated. The routine also locates the position of these phases with respect to a barrier of the controller. A barrier must be crossed on both rings concurrently.

Synopsis

The routine first makes the distinction between a single ring and a dual ring controller. In the single ring case, the routine terminates the indicated phase and proceeds to activate the next phase if there is no overlapping phase. If there is an overlapping phase, the surrogate phase becomes the parent phase. Routine ASIG is subsequently called to update the link-specific signals to reflect the activation of the surrogate phase.

In the dual ring case, the processing is much more complex than that of the single ring case. The routine must first establish the positions of those phases that are to be terminated with respect to the barrier of the controller. If both phases are at a barrier, the routine then crosses the barriers to activate those phases that

were called. If both phases are not at a barrier, it implies that both phases that are to be activated must be either at the same side of a barrier with those phases that are terminating or at the other side of a barrier. Activation of the phases in different barrier boundaries is not permitted. While searching to activate new phases, rest-in-red and dual entry options are also considered.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Index denoting phase, set in do-loop from 1 to 4
IR	I	Hold Area	Ring of terminated phase
J	I	Hold Area	Phase on Ring 1 at controller
JJ	I	Hold Area	Status code for new surrogate phase
K	I	Hold Area	Phase on Ring 2 at controller
M	I	Hold Area	Surrogate phase
M1	I	Hold Area	First phase on other side of barrier

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
Il	I	Hold Area	Input, actuated node number

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT6/ NACNT

/ACT8/ NACDS

/ACT12/ NASTAT

/ACT16/ MTERM

5.66 Subroutine TPTOUT

Program Narrative

Purpose

TPTOUT writes out trajectory plot data to the plot data tape at the end of each trajectory plot output interval.

Assumptions

TPTOUT assumes that whenever it is called, it is time to output records to the plot data tape and also that a tape unit has been provided by the user.

Approach

TPTOUT loops over all freeway links specified by the user, and outputs records for all vehicles in all lanes of each of these links.

Synopsis

TPTOUT loops over all freeway links in the ITPLNK array specified by the user. For each link, TPTOUT loops over all lanes. In each lane, the farthest upstream vehicle, if any, is found, and the vehicle number and location are stored in the plot output buffer. Then, each successive leader in the lane is processed until all vehicles in the lane have been processed. When all vehicles in a lane have been processed, or whenever the buffer becomes full, the buffer is written out to the tape, with continuation records if necessary.

Models and Algorithms

For a description of the plot data tape format, see the INTRAS Final Report.

Date, Programmer

April, 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	ITPLNK array index, set in do- loop from 1 to 24
ICOD	I	Switch	Auxiliary lane type code (0,1,2) for (acceleration, deceleration, both)
ILOC	I	Hold Area	Vehicle position
IV	I	Hold Area	Vehicle number
IVL	I	Hold Area	Leader vehicle number
J	I	Do-loop Index	Lane number index, set in do- loop from 1 to 9
JBUF(101)	I	Hold Area	Plot output buffer
JJ	I	Hold Area	Link array parameter number of farthest upstream vehicle
JP	I	Counter	Number of entries in plot out- put buffer
KK	I	Switch	(1,2) for (first, second) auxiliary lane
L	I	Hold Area	Link number
LDII	I	Hold Area	Location of beginning of de- celeration auxiliary, if any, 0 otherwise
LENA	I	Hold Area	Auxiliary lane length
LENG	I	Hold Area	Link length
NAUX1	I	Hold Area	Lane number of first auxiliary lane

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>

NAUX2	I	Hold Area	Lane number of second auxiliary lane

NLANE	I	Hold Area	Number of thru lanes on link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IFN, IFORM, ILEN, IOP, ITPLNK

/IPLTN/ IPLTN

/VBUF/ NF

/SIZE/ MXLFR

5.67 Subroutine TSIG

Program Narrative

Purpose

TSIG examines the signal code and decides whether it is prohibitive or permissive for a given vehicle, whether it just became permissive, and whether it is a stop, yield or right-turn-on-red signal.

Assumptions

TSIG assumes that the vehicle and link environment areas contain all information pertaining to the given vehicle.

Approach

The signal code facing the given approach is examined, and a branch is taken according to the code to determine if it is permissive or prohibitive.

Synopsis

The signal code and vehicle turn code are found. A branch is taken according to the signal code to decide whether it is prohibitive: if the code is for green, then it is permissive. If it is always green, then processing is over; otherwise, TSIG will check whether it was permissive during the last time step. If the signal is red, then it is prohibitive, and TSIG must check whether it has been prohibitive for NPRHB time steps. If the signal is green for specified movements only, then the turn code must be compared and the signal is treated as red or green accordingly. If the signal is amber or yield, then the number of intervals is examined; if there is only one, then this implies a yield sign. Flags are set accordingly and processing is over. If it is amber, a decision will be made as to whether it is prohibitive or permissive.

For all vehicles facing a prohibitive (red) signal for their movement, TSIG checks whether the signal was permissive during the last NPRHB time steps (this number is a constant). If it was, then the vehicle will be allowed to

go through the red light since, presumably, it could not stop in time.

For all vehicles facing a green or amber signal, TSIG checks whether the signal was permissivle during the previous time step, so that the JSIGC flag can be set. If the previous signal was prohibitive and the current signal is amber, then the amber is assumed to be prohibitive. If the previous signal was permissive and the current signal is amber, then further decisions must be made: if the vehicle is a left-turner and first in queue, then the signal is permissive. If the vehicle is already stopped, then it is prohibitive. If the vehicle is at the stopline, however, the signal is considered permissive. Finally, if the vehicle must decelerate at an unacceptable rate in order to stop, then the signal is considered permissive.

Finally, if the signal code indicates a stop sign and the vehicle has already stopped, then the signal is permissive; otherwise, it is prohibitive. If the signal indicates right-turn-on-red, and the vehicle is a right-turner which has already stopped, then the signal is permissive; otherwise, it is prohibitive.

Models and Algorithms

None.

Date, Programmer

January, 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	SIGI array index, set in do-loop from 1 to 4
IA	I	Hold Area	Deceleration required to stop at stopline
IDC	I	Hold Area	Driver type code

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IDIS	I	Hold Area	Distance from stopline
II	I	Hold Area	SIGI index of given link
IN	I	Hold Area	Current interval number
IN1	I	Hold Area	Previous interval number
ISGN	I	Hold Area	Previous signal code
ISIG	I	Hold Area	Current signal code
IT	I	Hold Area	Time remaining in current interval
IVT	I	Hold Area	Vehicle type code
L	I	Hold Area	Link number and link type, packed
M	I	Hold Area	Magnitude of packed previous signal location
MM	I	Hold Area	M/10
NPRHB	I	Constant	Number of previous time steps checked for permissive signal if current signal prohibitive; set equal to 3 in data statement

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A8/IAMBR, IPACK, MINAC

/S2/IDTS

/LENVI/IOOF, JDNODE, LANE, LINK, LTSURF

/VENVI/JSIGC, JSIGP, JSIGS

/SRFLAG/LTFLAG

/VBUF/NF

/A6/NPHS, SIG, SIGT

/A3/SIGI

5.68 Subroutine TSTSAT

Program Narrative

Purpose

TSTSAT tests for saturation in the receiving link and decides whether or not a given vehicle may discharge.

Assumptions

TSTSAT assumes that the link and vehicle environment areas contain all information pertaining to the vehicle in question.

Approach

The vehicle is processed according to its turn code and whether it is entering a freeway link. If there is too much spillback, the vehicle cannot discharge; if at least one lane is acceptable, it will discharge as far as TSTSAT is concerned.

Synopsis

If the vehicle is entering an exit link, then it will automatically be allowed to discharge. If not, then it is processed according to its turn code and type of receiving link, as follows (in all cases, if at least one lane is acceptable, then the vehicle will be allowed to discharge):

- (a) Thru vehicle discharging to non-freeway link:
If in normal mode, TSTSAT examines the receiving lane and lanes to the immediate right and left, if any. If in look-ahead mode, only the receiving lane is examined. TSTSAT loops over the necessary lanes, checking each for spillback. If there is none, or if the upstream vehicle is accelerating, or if it will clear the stopline before halting, then the spillback is acceptable. Otherwise, it is decided whether to discharge into spillback by comparing the driver type code against the probability of joining

spillback of a given number of cars. If spillback is acceptable, the leader position after emergency deceleration must still be compared with the given vehicle's position after normal deceleration -- the two must avoid collision.

(This test assures consistency with LANES requirements.) Finally, if the lane being examined is to the right or left of the receiving lane, then the right or left follower must have adequate stopping distance if the given vehicle were to become its new leader.

- (b) Left-turning vehicle discharging onto non-freeway link: All thru lanes on the receiving link are examined. TSTSAT loops over all lanes, checking for spillback. (If there is a lane with no upstream vehicle, then the given vehicle will discharge). The position of the leader after processing is computed. The speed of the given vehicle when entering the new link is found by extrapolating backwards, as well as the time to be spent on the new link. Then the final position if decelerating at 14 fpss is found and compared with final leader position; if the two will not collide, then the lane is acceptable.
- (c) Right-turning vehicle discharging onto non-freeway link: Only the rightmost lane in the receiving link is examined. The check is similar to left-turners -- if the vehicle can avoid collision with the leader by decelerating at 7 fpss, then the lane is acceptable.
- (d) Vehicles discharging onto freeway link: The vehicle located in the freeway link receiving lane closest to the discharging vehicle is identified. This identification is accomplished by reviewing the contents of the LNKF array to determine the furthest upstream vehicle at the end of the last freeway time step. The IORG array is then searched to determine if additional vehicles have discharged into the subject lane in the interim. If there is room for the given vehicle to clear the stopline without colliding with the furthest upstream freeway vehicle, then the lane is acceptable for discharge.

Models and Algorithms

None.

Date, Programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Receiving lane number, set in do-loop from I1 to I2
IDC	I	Hold Area	Driver type code
IHM	I	Hold Area	Minimum leader horizon (set to dummy value)
II	I	Hold Area	Lane number for debug printout
INTL	I	Hold Area	Intersection width
IOFST	I	Hold Area	Receiving lane offset relative to upstream lane (ramps only)
IPOS	I	Hold Area	Position of rear of upstream vehicle after processing
IREC	I	Hold Area	Receiving lane of vehicle's current lane
ISPX	I	Hold Area	Position of right-turner on entry onto new link
IT	I	Hold Area	Time step for freeway vehicle to travel
ITC	I	Switch	(0,1,2) for (left, thru, right) vehicle

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IV	I	Hold Area	Vehicle number of vehicle waiting to discharge onto freeway
IVT	I	Hold Area	Vehicle type code
IX	I	Hold Area	LLOR index of leftmost lane
IXDIS	I	Hold Area	Distance travelled on new link while decelerating
IXRAD	I	Hold Area	Temporary storage in solution of equation for time spent on new link
IXTIM	I	Hold Area	Time spent on new link
IXVEL	I	Hold Area	Vehicle speed on entering new link
IZL	I	Hold Area	Leader position after emergency deceleration
IZS	I	Hold Area	Vehicle's stopping distance on new link
I1	I	Hold Area	Rightmost lane to be examined
I2	I	Hold Area	Leftmost lane to be examined
J	I	Do-loop Index	LLOR index, set in do-loop from 1 to 5
JDS	I	Hold Area	Right or left follower's position after decelerating to halt
JJ	I	Hold Area	Receiving lane number for debug print
JSP	I	Hold Area	Left or right follower's speed
JSPX	I	Hold Area	Vehicle's position on new link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
KK	I	Hold Area	Vehicle position if discharged into spillback
LD	I	Hold Area	Downstream freeway link
LLO	I	Hold Area	Lane number unpacked from LLOR array
LND	I	Hold Area	Downstream freeway lane number
LPOS	I	Hold Area	Predicted final position of vehicle
NBUF	I	Counter	Number of vehicles already waiting to discharge onto freeway
NLANE	I	Hold Area	Number of lanes in receiving link
XRAD	Real	Hold Area	Discriminant in solving quadratic equation for time spent on new link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IFLAG	I	Switch	Input, 0 if normal mode, vehicle ready to discharge this time step; 1 if vehicle will not discharge but must look ahead in receiving lane to decide whether to begin decelerating
ISAT	I	Switch	Output, (0,1) if vehicle (will, will not) discharge this time step

Common Blocks and Variables Referenced

/A8/ CLEN, IPACK, SPLPCT

/S2/ IDTS, ITIME, ITIMEF

/ERTRAN/ IEPAR, IER, NPAR

/LENVI/ IOOF, IRALL, LACR, LANE, LINK,
LLNA, LLNR, LLOR, LPOR, LPRR, LSPR,
LUVR, LVL R

/ONVEH/ IORG, IRV, NORG

/VENVI/ JPFOL, JV FAC, JVFS, JV FSP

/SRFLAG/ LTFLAG

/VBUF/ NF

5.69 Subroutine TYPE

Program Narrative

Purpose

TYPE generates a new freeway vehicle type.

Assumptions

TYPE assumes that the input parameters (percent distribution of freeway vehicles by type and number of cells in the distribution) have been primed correctly.

Approach

The new vehicle type is determined from a random number.

Synopsis

First a random number is generated. If it is less than or equal to the percent of type 1 vehicles, the new vehicle will be a type 1 vehicle. If the random number is greater than the percent of type 1 vehicles and less than or equal to the sum of percents of types 1 and 2 vehicles, the new vehicle will be a type 2 vehicle. This procedure is applied to the remaining types (by testing against the sum of their corresponding percents) until the vehicle type has been determined.

Models and Algorithms

None.

Date, Programmer

June 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Vehicle type index. Set in do-loop from 2 to II
ICS	I	Hold Area	Sum of percents in distribu- tion
MR	I	Dummy argu- ment to RANDOM	
NR	I	Hold Area	Random number
RN	R	Dummy argu- ment to RANDOM	

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
II	I	Hold Area	Input, number of cells in percent distribution of free- way vehicle types
ITYP	I	Hold Area	Output, new vehicle type
JTEMP(10)	I	Hold Area	Input, percent distribution of freeway vehicle types

Common Blocks and Variables Referenced

None.

5.70 Subroutine UPACT

Program Narrative

Purpose

To serve as an executive routine to update the status of an active phase. The active phase can either be actuated or non-actuated.

Assumptions

UPACT assumes that the active phase can be either actuated or non-actuated.

Approach

This routine acts as an executive routine for updating the status of a current active phase. The active phase can either be actuated or non-actuated. If it is actuated, depending on the status code of this actuated phase, GRNSIG or REDSIG is then called in to update its status. If it is non-actuated, test if past the yield point. If so, call FZCL to determine if another phase is called. If it is, set decision flag to terminate current phase. After updating the status for all active phases on both rings (if it was a dual ring controller), routine DECACT is called to determine whether the current active phase should be terminated. Finally, TERMFZ is called to act on the termination flag set by the DECACT routine. If the flag is set, the current phase(s) is terminated and the subsequent phase is activated. If not, the current active phase remains active to terminate the indicated phase and at the same time to activate the new one(s).

Synopsis

The routine loops over rings to identify the active phase, NP, on each ring. The active phase can either be actuated or non-actuated.

Routine GRNSIG is called if the status code of actuated phase, NP, is less than, or equal to, 3 to process the initial, added initial and green interval for phase, NP. Routine REDSIG is called if the status code of actuated phase, NP, is equal to 4, 5 or 6 to process the amber, red clearance and red revert interval for phase, NP.

If phase, NP, is non-actuated, there will be no action taken on phase, NP, as long as the phase-specific timer is outside the yield interval. For coordinated controllers, the duration of the yield interval is specified. For non-coordinated controllers, the yield interval extends indefinitely beyond the yield point. If the timer is within the yield interval and the signal servicing phase, NP, is still green, FZCL is called to determine whether any other phase on this semi-actuated controller has been called. If there has been a call, UPACT will set the flag to indicate that the phase, NP, is subject to terminate. Otherwise, phase, NP, will remain as it is through this time-step. If the timer is within the yield interval and the signal is amber, routine ASIG is called to set the link-specific signal code from green to red directly after half the amber duration has elapsed. This is done because there is no movement-specific amber signal code. The final termination flag for phase, NP, will not be set until the amber interval and red clearance interval, if any, are completely exhausted.

Models and Algorithms

None.

Date, Programmer

May 1977. Meena Nagarajan.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Index, set in do-loop from 1,8 to reset array of termination codes
ICC	I	Hold Area	Coordination code
IRNGS	I	Hold Area	Number of rings
JJ	I	Hold Area	Status code of phase NP
K	I	Hold Area	Non-actuated phase
KK	I	Hold Area	Status code
MTMR	I	Hold Area	Timer

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I1	I	Hold Area	Input, actuated node number

Common Blocks and Variables Referenced

/ACT1/ NAFZ

/ACT2/ NAAFZ

/ACT3/ NATMR

/ACT8/ NACDS

/ACT9/ NACYC

/ACT11/ NADEC

/ACT12/ NASTAT

/ACT16/ MTERM

5.71 Subroutine UPSIG

Program Narrative

Purpose

UPSIG updates the signal settings after each simulation time-step, flags the last vehicle in queue whenever a signal turns green, and also checks for cycle failures.

Assumptions

None.

Approach

UPSIG loops over all nodes in the network, skipping those under sign control and processing those with actuated controls separately from those with fixed-time signals.

Synopsis

UPSIG loops over all nodes in the network; at each node the intersection control is examined. If it is an actuated control, then one of nine special routines must be called to update the signal, depending on the user-specified algorithm. If it is a sign control, the node is ignored.

Otherwise, it is assumed to be a fixed-time signal and the time remaining in the current interval is updated and checked. If the interval is not over, then processing goes on to the next node. If it is over, then the SIFT array is initialized with the new signal code and time remaining (i.e., interval duration). Then UPSIG loops over all approaches to the node and stores the new signal facing each approach in the appropriate link array. If the signal just turned green this time-step for a given approach, then each lane of the approach link must be examined. If there are any vehicles in the lane and there is a queue, then the last vehicle in queue is flagged (for future use in checking for cycle failure).

If the signal just turned red and the previous signal

was green (or if it was amber and green before that), then each lane must be examined for cycle failure; that is, if any vehicles were in queue when the signal turned green and have not discharged, then a cycle failure has occurred, and the number of cycle failures must be incremented and stored in the appropriate link array.

When all nodes have been processed, control returns to the calling routine.

Models and Algorithms

None.

Date, Programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-Loop Index	Node number, set in do-loop from 1 to NTOTN
IAUX	I	Hold Area	Auxiliary lane flag from LASTV (not used)
ICF	I	Switch	(0,1) if first cycle failure (has not, has) been found
IL	I	Hold Area	Leader vehicle
IQ	I	Switch	Queue code = (0,1,2) for (no queue, queue, cycle failure)
ISG	I	Hold Area	Previous signal code
ISGN	I	Hold Area	New signal code
IT	I	Switch	Link type = (1,2) for (ramp, surface) link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IV	I	Hold Area	Farthest upstream vehicle
IVL	I	Hold Area	Previous interval number
IVLN	I	Hold Area	New interval number
J	I	Do-Loop Index	Approach number
JCF	I	Counter	Total number of cycle failures on link
K	I	Do-Loop Index	Lane number, set in do-loop from 1 to NLANE
KA	I	Hold Area	Interval and time remaining in interval (packed)
KB	I	Hold Area	Previous signal code
KC	I	Hold Area	New signal code
L	I	Hold Area	Link number of approach link
LANTYP	I	Hold Area	Lane type, output from LASTV (not used)
M	I	Hold Area	Packed signal codes
M1	I	Hold Area	Magnitude of digit position in M beyond new signal code
M2	I	Hold Area	Magnitude of digit position of new signal code
N	I	Switch	Actuated control algorithm number, user-specified from 1 to 9
NLANE	I	Hold Area	Number of lanes in link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/S2/ IDT, IDTF, IDTS, IFDT, ITIME, ITIMEF,
MCYCL

/SIZE/ IS

/A6/ NPHS, SIG, SIGT

/A3/ NTOTN, SIGI

5.72 Program LOCON

Program Narrative

Purpose

LOCON is the main supervisor for the low activity overlay routines. It performs a variety of functions depending on the value of a flag set by the calling program.

Assumptions

None.

Approach

A branch is taken according to the value of the ISSEQ flag, and the appropriate code is executed to perform the desired functions.

Synopsis

The following describes the functions performed for each value of ISSEQ:

(a) ISSEQ=1, beginning of fill time or new subinterval. INIT is called to initialize various flags and counters. The number of fill time time steps is computed. SETCON primes traffic control at any actuated nodes. LRANK orders the freeway links from downstream to upstream. FTSC calculates the freeway time step. The arrays for number of vehicles processed and number making each turn movement on each link are primed.

(b) ISSEQ=2, possible end of fill time: FILTST is called to decide whether equilibrium has been attained. If not, fill time continues; if so, the length of fill time is computed, and processing continues with (c) below.

(c) ISSEQ=3, start of first subinterval. RESET is called to reset any parameters which may have changed during fill time. MXINIT is set to zero to indicate fill time is over, and NIDT is restored to the number of time steps in the subinterval. Then processing continues as for (a), beginning with a call to FTSC.

(d) ISSEQ=4, incident detection: if on-line incident detection is desired, and it is time for incident detection, SINCES is called to perform the incident detection functions.

(e) ISSEQ=5, contour plot data: CPTOUT is called to output contour plot data to tape.

(f) ISSEQ=6, standard output report: CYCP is called to print the standard output report.

(g) ISSEQ=7, detailed report: INTST is called to output the detailed report.

Models and Algorithms

None.

Date, Programmer

January, 1976. Dave Wicks.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-Loop Index	Link index, set in do-loop from 1 to MXLFR, MXLRA or MXLSU
J	I	Counter	Absolute link number
KK	I	Hold Area	Temporary storage for unpacked link parameters

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/S2/ IDT, IDTF, IDTS, IFDT, IPSL, ITIME,
ITIMEF, NIDT

/FORINC/ INCTIM, LINEON

/S3/ ISRET, ISSEQ

/CRASH/ IUPDAT

/AID/ LTRNV, MNUVR, RTRNV, THRUV, VCONT,
VPROC

/CONTRL/ MINSTS, MXINIT, NTRN

/SIZE/ MXLFR, MXLRA, MXLSU

/NIDTSV/ NIDTSV

5.73 Subroutine CPTOUT

Program Narrative

Purpose

CPTOUT writes contour plot data records (type 4) to the plot data tape.

Assumptions

CPTOUT assumes that it is time to write out the type 4 records and that tape unit 19 has been provided by the user.

Approach

CPTOUT loops over all links, processing all detectors in all lanes on each link.

Synopsis

CPTOUT loops over all links in the ICPLNK array. For each link, the first detector is found, and successive detectors are examined until the first plot detector is found. Then spot speed and volume are accumulated for all plot detectors in different lanes at the same location on the link. When all detectors at a given location have been processed, the average spot speed and volume are stored in the output buffer. Processing goes on to the next set of detectors until all plot detectors have been processed. If the buffer becomes full before all have been processed, then a record is output to tape with a continuation flag to indicate that a second record follows for the same link. Otherwise, one record per link is output.

Models and Algorithms

See the INTRAS final report for a complete description of the plot data tape.

Date, programmer

April, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	ICPLNK array index, set in do-loop from 1 to 24
ISPTSP	I	Hold Area	Sum of spot speeds for all vehicles crossing station
IVOL	I	Hold Area	Total vehicle volume at detector station
JBUF (101)	I	Hold Area	Plot output data buffer
JP	I	Counter	Number of entries in plot output data
KDET	I	Hold Area	Pointer to buffer detector being processed
KDETN	I	Hold Area	Pointer to next detector
L	I	Hold Area	Link number
NUMD	I	Counter	Number of detectors at given locations on link

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/A4/DTCTR

/CONTRL/ICPLNK,ICPOP,ICN,IFORM,ILEN,IOP

/IPLTN/IPLTN

/SIZE/MXLFR

5.74 Subroutine CYCP

Program Narrative

Purpose

CYCP prints out the standard output report of cumulative or subinterval-specific statistics.

Assumptions

CYCP assumes that it is time to print the report if it is called.

Approach

CYCP prints a table for ramp and surface links and then prints a separate table for freeway links.

Synopsis

First, clock time is computed and the appropriate non-freeway table heading is printed for either cumulative or subinterval-specific statistics. CYCP then loops over all ramp links and all surface links, accumulating and printing a line of data for each link as follows: all the parameters needed are unpacked by calling the appropriate service routine. The statistics are converted to real values, converted to units desired for printout and computed or adjusted according to whether statistics are cumulative or subinterval specific. One line is printed giving the link number, upstream and downstream nodes, and all statistics identified by the table heading.

As each ramp and surface link is processed, statistics are accumulated which relate to the total of all non-freeway links. These statistics are printed at the end of the table.

Similarly, a table is printed with statistics for each freeway link. At the end are total statistics for all freeway links, and, finally, statistics for all freeway and non-freeway links together.

Finally, if there are any freeway link stations, a table of statistics gathered by these stations, giving a

breakdown of traffic speed and headway at each station, is printed.

Models and Algorithms

None.

Date, Programmer

January, 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
A	R	Hold Area	Temporary storage for computing average speed and headway
ALANE2	R	Hold Area	Number of thru lanes
AMOVSV	R	Hold Area	Link surface area
AUX1L2	R	Hold Area	1/3 length of first auxiliary lane
AUX2L2	R	Hold Area	1/3 length of second auxiliary lane
AVGCON	R	Hold Area	Average link contents
B	R	Hold Area	Headway breakdown category
D	R	Hold Area	Delay time per vehicle-mile
DELSV	R	Hold Area	Delay time
DELVM	R	Hold Area	Vehicle-minutes/vehicle-mile (delay)
DFNSITY	R	Hold Area	Vehicle density
DIS	R	Hold Area	Link length

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Identification, and Modification</u>
DLY	R	Hold Area	Delay time
HDLAY3	R	Hold Area	Delay time, halves-of-an-hour
HDSEC	R	Hold Area	Mean headway
HEADY (14)	R	Hold Area	Headway categories for data station table
HMOVE3	R	Hold Area	Move time, halves-of-an-hour
I	I	Hold Area	Total delay time
IAVCON	I	Hold Area	Average contents
IAVGST	I	Hold Area	Total average saturation
IAVSAT	I	Hold Area	Average saturation percentage
ICYCF2	I	Hold Area	Number of cycle failures
IDLY	I	Hold Area	Delay time
IFLAG	I	Switch	(1,2) if (ramp, surface) links being processed
IHAF	I	Hold Area	Temporary storage for un- packing link parameters
IHO	I	Hold Area	Clock time, hours
IHSDL2	I	Hold Area	Stopped delay, halves-of-an- hour
IMO	I	Hold Area	Clock time, minutes
IPSDLY	I	Hold Area	Percentage stopped delay
IROUND	I	Hold Area	Rounding factor for rounding from real to integer

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IRPKT2	I	Hold Area	Right pocket lane number
ISDLAY	I	Hold Area	Stopped delay time
ISO	I	Hold Area	Clock time, seconds
ISPEED (13)	I	Hold Area	Speed categories for data station table
ISR	I	Hold Area	Temporary storage for converting simulation time to clock time
ISSDL2	I	Hold Area	Stopped delay, seconds
ISUM	I	Hold Area	Sum of speeds or headways
ITAREA	I	Hold Area	Total surface area for all non- freeway links
ITCC	I	Hold Area	Total vehicle contents for all links
ITCCON	I	Hold Area	Total vehicle contents for all non-freeway links
ITCYCF	I	Hold Area	Total number of cycle failures
ITLCHG	I	Hold Area	Total number of lane changes
ITPSDL	I	Hold Area	Percentage stopped delay
ITSTOP	I	Hold Area	Stopped delay
IVIN	I	Hold Area	Vehicles entering link during subinterval
J	I	General pur- pose do-loop index	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
K	I	General purpose do-loop index	
KDN	I	Hold Area	Downstream node of link
KHDWY	I	Hold Area	Headway of vehicles passing data station
KK	I	Hold Area	Pointer to next LHCELL word to be filled
KKK	I	Hold Area	Index of highest LHCELL category to be printed
KLD	I	Hold Area	Link number of data station link
KLOC	I	Hold Area	Location of data station
KSPED	I	Hold Area	Speed of first non-zero speed cell
KUP	I	Hold Area	Upstream node of link
LANE2	I	Hold Area	Number of thru lanes
LDIV	I	Hold Area	Dividing factor for unpacking speed and headway categories
LFPKT2	I	Hold Area	Left pocket lane code
LHCELL (7,20)	I	Hold Area	Headway categories for each lane for data station table
LHCONT (7)	I	Hold Area	Number of headway inputs for each lane
LKAREA	I	Hold Area	Link surface area
LMUL	I	Hold Area	Multiplicative factor for unpacking speed and headway categories

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definiton, Initialization, and Modification</u>
LNSTAT (7)	I	Hold Area	Lane nodes for all lanes in link
LSCELL (7,20)	I	Hold Area	Speed categories for each lane for data station table
LSCONT (7)	I	Hold Area	Number of speed inputs for each lane
M	I	Hold Area	Number of ramp or surface links
MULT	I	Hold Area	Speed cell scale factor
N	I	Hold Area	Absolute link number
RATMT	R	Hold Area	Ratio of move time to total time
SDLAY3	R	Hold Area	Delay time, tenths of a second
SMOVE3	R	Hold Area	Move time, tenths of a second
SPEED	R	Hold Area	Average speed
SPMPH	R	Hold Area	Average speed
SUM	R	Hold Area	Sum of all speed categories
T	R	Hold Area	Time interval in question, i.e., since beginning of simulation or current subinterval
TAREA	R	Hold Area	Total link area
TAVC	R	Hold Area	Average contents for all links
TAVGCN	R	Hold Area	Average contents for all non-freeway links
TDEN	R	Hold Area	Average density for all free-way links

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
TDLAY	R	Hold Area	Delay time for all non-freeway links or all freeway links
TDLY	R	Hold Area	Delay time for all links
TOTSV	R	Hold Area	Travel time
TOTVM	R	Hold Area	Vehicle miles
TRATMT	R	Hold Area	Ratio of move to total time for all freeway or non-freeway links
TSPEED	R	Hold Area	Average speed for all freeway or non-freeway links
TVMDLY	R	Hold Area	Veh-min/veh-mile (delay) for all freeway or non-freeway links
TVMILE	R	Hold Area	Vehicle miles for all freeway or non-freeway links
TVMIN	R	Hold Area	Vehicle minutes for all freeway or non-freeway links
TVML	R	Hold Area	Vehicle miles for all links
TVMN	R	Hold Area	Vehicle minutes for all links
TVMTOT	R	Hold Area	Total veh-min/veh-miles
TVOL	R	Hold Area	Total volume
VMILES	R	Hold Area	Vehicle miles
VMIN	R	Hold Area	Vehicle minutes
VOLUME	R	Hold Area	Volume on link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS, IVFEET

/FREDAT/ FRCONT, FRHDWY, FRSPED

/A8/ IPACK

/S2/ IDT, IDTS, IHOURS, IMINS, ISECNS, ITIME,
NIDT

/A11/ ISTATN, LSTATN

/CONTRL/ ISTFLG

/SIZE/ MXLFR, MXLRA, MXLSU, NLFR, NLRA

5.75 Subroutine FILTST

Program Narrative

Purpose

FILTST decides whether equilibrium has been reached in the network and, hence, whether fill time is over.

Assumptions

None

Approach

FILTST checks whether equilibrium has been reached for two consecutive evaluation periods (i.e., less than 10% difference between vehicles entering and leaving the system).

Synopsis

FILTST loops over all freeway and surface entry links and counts the total number of vehicles on all entry links. Then the number of vehicles entering the system during the current evaluation period is computed. If none have entered, fill time will continue. Otherwise the percentage difference between vehicles leaving and entering the system is computed. If it is less than 10% and was also less than 10% during the previous evaluation period, fill time is over; otherwise, fill time continues.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Link index, set in do-loop from 1 to MXLFR, or MXLSU
II	I	Hold Area	Absolute link number
IPD	I	Hold Area	Percent difference between vehicles entering and leaving network
NDV	I	Hold Area	Number of vehicles discharged this evaluation period
NE	I	Hold Area	Number of vehicles on link
NENTR	I	Counter	Total number of vehicles on entry links
NEV	I	Hold Area	Number of vehicles entering system this evaluation period

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IFL	I	Switch	Output, (0,1) if fill time (is not, is) over

Common Blocks and Variables References

/A2/ENDNDS,NTOTL

/SIZE/MXLFR,MXLSU

/NFILTS/NDSCHG,NDSCHO,NENTRO,NLASEQ

/SIZE/NLFR,NLRA

5.76 Subroutine FTSC

Program Narrative

Purpose

FTSC computes the freeway and non-freeway simulation time steps

Assumptions

None

Approach

The freeway time step is determined according to the maximum vehicle density as well as the incoming demand.

Synopsis

The default time steps are set at one second. If the user has specified a time step, this value is used for both freeway and non-freeway movement. If no user-defined value was specified, and simulation has gone beyond the first subinterval, the freeway time step may be reset as follows: if no freeway link has a density of more than 30 vehicles per lane-mile, and no freeway entry link has a volume of more than 1500 vehicles per lane per hour, the freeway time step is increased from one second to two seconds.

Finally, the number of surface time steps for the next subinterval (or for fill time) is computed.

Models and Algorithms

None

Date, programmer

January, 1976. Dave Wicks

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Freeway link index or entry link index
IHAF	I	Hold Area	Number of lanes
II	I	Hold Area	Freeway link density
ITSBUF (45)	I	Hold Area	Freeway link parameter vector
J	I	Hold Area	Entry link number
K	I	Hold Area	Vehicle volume x 5280
MDENS	I	Hold Area	Maximum density on any freeway link

Arguments Passed to Subroutine

None

Common blocks and variables referenced

/S2/IDTF,IDTS,ITIME,NIDT

/CONTRL/IRSCT,MINSTS,MXINIT,NDT

/SIZE/MXLFR

/A7/NS,SNODE

5.77 Subroutine INIT

Program Narrative

Purpose

INIT initializes all flags, counters and other parameters necessary at the beginning of each subinterval.

Assumptions

None

Approach

Initialization is performed as necessary for freeway, ramp, and surface link parameters.

Synopsis

First, if there are any freeway links, the freeway data station statistics arrays are zeroed out. If statistics are to be subinterval-specific, various counter and accumulators in the freeway link array are cleared.

If there are any ramp links and statistics are subinterval-specific, various ramp link counters and accumulators are cleared.

If there are any surface links, then various surface link counters and accumulators are cleared, some only if statistics are subinterval-specific, others in all cases.

Finally, the FUEL data buffer pointer is zeroed out. The number of feet travelled on each link is cleared if statistics are subinterval-specific.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Link index, set in do-loop from 1 to MXLFR, MXLRA, or MXLSN
ITMP	I	Hold Area	Temporary storage for zeroing out link para- meters
J	I	Do-loop index	Link parameter index
K	I	Do-loop index	Lane index, set in do-loop from 1 to 5

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/FREDAT/FRCONT,FRFLAG,FRHDWY,FRSPED

/CONTRL/ISTFLG

/A2/IVFEET

/NRGY/JVACCP

/SIZE/MXLFR,MXLRA,MXLSU

5.78 Subroutine INTST

Program Narrative

Purpose

INTST produces the detailed simulation output report.

Assumptions

INTST assumes, whenever it is called, that it is time to print the detailed report.

Approach

INTST unpacks all relevant data from the appropriate link array and prints one line for each link.

Synopsis

INTST loops through the same code three times, once each for freeway, ramp and surface links. On each pass, INTST loops over all links of the given type. For each of these links, the appropriate unpacking routine is called to retrieve all required data. The number of vehicles in each lane must be computed. Finally, parameters common to all link types are computed, with consideration given to whether statistics are subinterval-specific or cumulative. One line in the table is printed for each link, giving link type, upstream and downstream nodes, and all the parameters just unpacked and computed.

Models and Algorithms

None.

Date, Programmer

April, 1976. Barbara Andrews.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
AVGSP	R	Hold Area	Average speed
DLAYH	R	Hold Area	Delay time, halves-of-an-hour
DLAYT	R	Hold Area	Delay time, tenths-of-a-second
DLYVEH	R	Hold Area	Delay time per vehicle
DTIME	R	Hold Area	Delay time
I	I	Switch	(1,2,3) for (freeway, ramp, surface) vehicles
IDLY	I	Hold Area	Percentage stopped delay
IHO	I	Hold Area	Clock time, hours
II	I	Do-loop Index	Lane index, set in do-loop from 1 to 9
IMO	I	Hold Area	Clock time, minutes
INDEX	I	Hold Area	Absolute link number
ISO	I	Hold Area	Clock time, seconds
ISR	I	Hold Area	Temporary storage for computing clock time
IVL	I	Hold Area	Number of vehicles turning left
IVR	I	Hold Area	Number of vehicles turning right
IVT	I	Hold Area	Number of thru vehicles
J	I	Do-loop Index	Lane index, set in WRITE state- ment from 1 to 9

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
K	I	Do-loop Index	Vehicle index, set in do-loop from 1 to MXVFR, MXVRA or MXVSU
L	I	Hold Area	Link Number
LMX	I	Hold Area	Number of links of type being processed
LTYPE (3)	I	Constants	Hollerith labels for link types, set in DATA statement
NLANE (9)	I	Counters	Number of vehicles in each lane of link
RMOVEH	R	Hold Area	Move time, halves-of-an-hour
RMOVET	R	Hold Area	Move time, tenths-of-a-second
SDLYH	R	Hold Area	Stopped delay, halves-of-an- hour
SDLYT	R	Hold Area	Stopped delay, tenths-of-a- second
VDIS	R	Hold Area	Number of vehicles discharged
VMILE	R	Hold Area	Vehicle-miles
VMIN	R	Hold Area	Vehicle-minutes

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS, IVFEET

/S2/ IHOURS, IMINS, ISECNS, ITIME

/A8/ IPACK

/CONTRL/ ISTFLG

/A10/ LTRNV, MNUVR, RTRNV, THRUV

/SIZE/ MXLFR, MXLRA, MXLSU, MXVFR, MXVRA,
MXVSU, NLFR, NLRA

5.79 Subroutine RESET

Program Narrative

Purpose

RESET resets all counters and accumulators necessary at the end of fill time, in preparation for the first subinterval.

Assumptions

None

Approach

All link parameters are cleared, then all other parameters including those for detectors and for the FUEL module.

Synopsis

First, all freeway link data station statistics are cleared. Then various freeway link statistics are zeroed out. Next, various ramp and surface counters and accumulators are zeroed out. Accumulators for various output intervals must be cleared, and the simulation time must be reset to zero.

Certain detector statistics are reset to zero. The FUEL buffer pointer is cleared, and the temporary FUEL data unit (22) is rewound.

Models and Algorithms

None

Date, programmer

January, 1976. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General-purpose do-loop index	
IB	I	Dummy argument to DRWS	
IPC	I	Switch	(0,1) if detector (is not, is) a plot detector
ITMP	I	Hold Area	Temporary storage of parameter to be packed
ITP	I	Switch	(0,1) if detector (is, is not) doppler radar
J	I	Do-loop index	Link parameter index
K	I	Do-loop index	Lane index, set in do-loop from 1 to 5

Arguments passed to subroutine

None

Common Blocks and Variables Referenced

/A4/DTCTR, IDET

/FREDAT/FRCONT, FRFLAG, FRHDWY, FRSPED, FRTIM

/S2/ICOINT, ICYCPO, IEXCS, IMCYCL, ITIME, ITIMEF, ITRINT

/CONTRL/IFN, IOP, MXINIT

/A2/IVFEET, NTOTL

/CFUEL/JFUEL

/NRGY/JVACCP

/A10/MNUVR

/SIZE/MXLFR,MXLRA,MXLSU

/A11/NSTATN

5.80 Subroutine SINCl

Program Narrative

Purpose

SINCl is a routine designed to search for freeway incidents using the "California Logic" (Ref. 1). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes three parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage.

The first two calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate.

On the third, and subsequent, call to SINCl, a loop over all detector stations used for incident detection is initiated. Three parameters are computed; F1KT, the change in occupancy between successive stations, F3KIT, the percent change in occupancy over time at the downstream station, and F2KT, the percent change in occupancy between successive stations.

These parameters are compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was the "California Algorithm," as described in Payne (Ref. 1).

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AK1	R	Hold Area	Occupancy threshold
AK2	R	Hold Area	Percent occupancy change threshold
AK3	R	Hold Area	Percent change in occupancy over time threshold
F1KT	R	Hold Area	Defined as a change in occupancy between successive stations
F2KT	R	Hold Area	Percent change in occupancy between successive stations
F3KIT	R	Hold Area	Percent change in occupancy over time at the downstream station of the pair under consideration
ICODE	I	Hold Area	Incident code
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IFLAG	I	Hold Area	Flag indicating last station in last station sequence (0,1) is (not last, last) station
IHEAD (2,4)	I	Hold Area	Holds alphanumeric data for output messages
I1	I	Hold Area	Time, seconds, at the start of the interval
I2	I	Hold Area	Time, seconds at the end of the interval
ICODE	I	Hold Area	Incident code
K	I	Hold Area	Detector station number
KALGM	I	Hold Area	Algorithm number, = 1
L	I	Do-loop Index	Loop from 1 to the number of freeway links
M	I	Hold Area	Present incident code
N	I	Hold Area	Node number
NODEDN	I	Hold Area	Downstream node number
NODE1	I	Hold Area	Upstream node number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ ENDNDS

/CONTRL/ IDRUN

/CRASH/ EPARAM, IFIN, INCODE, INPOS,
 INTVAL, IZERO, KALL, LOCATE,
 MAXSTA, OCCS, OCCOLD

/S2/ ITIMEF

/SIZE/ MXLFR

5.81 Subroutine SINC2

Program Narrative

Purpose

SINC2 is a routine designed to search for freeway incidents using an algorithm developed by Payne (Ref. 2). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes that six input parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage. On the first call to the routine, the incident detection frequency is tested and the run is aborted if it is found to be invalid.

The first two calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate.

On the third, and subsequent, call to SINC2, a loop over all detector stations used for incident detection is initiated. Four parameters are computed; F1, the change in occupancy between successive stations, F2, the percent change in occupancy over time at the downstream station, F3, the percent change in occupancy between successive stations, and F4, the occupancy at the downstream station.

These parameters are compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was developed by Payne (Ref. 2). Specifically it is Payne's Algorithm 8. It has been implemented with a user specified number of compression wave suppression periods.

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
F1	R	Hold Area	Defined as a change in occupancy between successive stations
F2	R	Hold Area	Percent change in occupancy over time at the downstream station of the pair under consideration
F3	R	Hold Area	Percent change in occupancy between successive stations
F4	R	Hold Area	The occupancy at the downstream station of the pair under consideration
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link
IHEAD (2,4)	I	Hold Area	Holds alphanumeric data for output messages
IST	I	Hold Area	Current incident status at given pair of stations
ISUPR	I	Hold Area	The number of compression wave suppression periods input
ITV	I	Hold Area	Incident updating interval in seconds
II	I	Hold Area	Time, seconds, at the start of the interval

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I2	I	Hold Area	Time, seconds at the end of the interval
JST	I	Hold Area	Interval incident code
K	I	Hold Area	Detector station number
KABORT	I	Hold Area	Abort flag
KALGM	I	Hold Area	Algorithm number, = 2
KIST	I	Hold Area	Flag to determine whether a tentative incident is considered an actual incident
L	I	Do-loop Index	Loop from 1 to the number of freeway links
M	I	Hold Area	Present incident code
N	I	Hold Area	Node number
NIST	I	Hold Area	Number of compression wave suppression periods plus 1
NNST	I	Do-loop index	Set from 1 to NIST
NODEDN	I	Hold Area	Downstream node number
NODE1	I	Hold Area	Upstream node number
T1	R	Hold Area	Occupancy threshold
T2	R	Hold Area	Occupancy threshold
T3	R	Hold Area	Occupancy threshold
T4	R	Hold Area	Occupancy threshold
T5	R	Hold Area	Occupancy threshold

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ENDNDS

/CONTRL/IDRUN

/CRASH/EPARAM,IFIN,INCODE,INPOS,INTVAL,
IZERO,KALL,LOCATE,MAXSTA,OCCS,OCCOLD

/S2/ITIMEF

/SIZE/MXLFR

5.82 Subroutine SINC3

Program Narrative

Purpose

SINC3 is a routine designed to search for freeway incidents using an algorithm incorporating the technique of double exponential smoothing (Ref. 3). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes that four input parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage. On the first call to the routine, the incident detection frequency is tested and the run is aborted if it is found to be invalid.

Up to the first four calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate, and to initialize the time series required for the exponential smoothing.

On subsequent calls to SINC3, a loop over all detector stations used for incident detection is initiated. One user specified parameter of two possible parameters is computed. The possible parameters are: FEAT(1), the change in occupancy between successive stations, and FEAT(2),

the percent change in occupancy over time at the downstream station.

The computed feature is compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was described in (Ref. 3).

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Hold Area	Exponential smoothing coefficient (between 0.0 and 1.0)
ALPHA	R	Hold Area	Absolute deviate smoothing coefficient (between 0.0 and 1.0)
AT	R	Hold Area	Temporary storage for smoothing computation
BT	R	Hold Area	Temporary storage for smoothing computation
ET	R	Hold Area	Difference between the predicted value and the actual value of the required feature

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
FEAT (2)	R	Hold Area	Array holding occupancy feature being used FEAT(1) Defined as a change in occupancy between suc- cessive stations FEAT(2) Percent change in occupancy between suc- sive stations
FEST	R	Hold Area	Predicted value of required feature
IFEAT	I	Hold Area	Requested feature (1 or 2)
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link
IFLAG	I	Hold Area	Flag indicating the last sta- tion in the last station sequence (0,1) for (not last, last) station
IHEAD (2,4)	I	Hold Area	Hold alphanumeric data for output messages
I1	I	Hold Area	Time, seconds, at the start of the interval
I2	I	Hold Area	Time, seconds, at the end of the interval
JABORT	I	Hold Area	Abort flag
JK	I	Hold Area	Incident code
K	I	Hold Area	Detector station number
KABORT	I	Hold Area	Abort flag
KALGM	I	Hold Area	Algorithm number, = 3

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KJTM	I	Hold Area	Number of times routine has been called
L	I	Do-loop Index	Loop from 1 to the number of freeway links
LABORT	I	Hold Area	Abort flag
M	I	Hold Area	Present incident code
MK	I	Hold Area	Incident code in previous time-step
N	I	Hold Area	Node number
NODEDN	I	Hold Area	Downstream node number
NODE1	I	Hold Area	Upstream node number
NSUPP	I	Hold Area	Number of algorithm suppression periods (1,4) of feature (1,2) is requested
SIGMA	R	Hold Area	Temporary storage used to compute initial standard deviate
T	R	Hold Area	Incident threshold
TAU	R	Hold Area	Time parameter. Equal to 1 for a 60 second update
ZT	R	Hold Area	Computed incident trigger

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ENDNDS

/CONTRL/IDRUN

/CRASH/EPARAM,IFIN,INCODE,INPOS,INTVAL,
IZERO,KALL,LOCATE,MAXSTA,OCCS,OCCOLD

/DBLEXP/SMTH1,SMTH2,SSQ,YERR,ABSD

/S2/ITIMEF

/SIZE/MXLFR

5.83 Subroutine SINCES

Program Narrative

Purpose

SINCES acts as an executive routine for the on-line incident detection option. It determines which incident detection algorithm is to be called and initiates the call.

Assumptions

The routine assumes that the on-line detector data file is available.

Approach

SINCES first positions the detector data file. It then calls a routine to gather the occupancy data. Finally, it determines which algorithm is to be called, and calls it.

Synopsis

On the first call to SINCES the routine initializes various incident detection COMMON storage elements. Various data arrays are initialized.

On all calls to the routine, the incident detection time counters are updated. At this point an endfile is written on the on-line detector data file and the file is rewound. Routine SPOINT is then called to accumulate the detector specific and station specific occupancy data. The on-line detector data file, no longer needed to rewind to position it for the next updating time period.

With occupancy data now stored in the proper COMMON arrays, SINCES then determines which of the three possible incident detection algorithms is to be implemented. That routine is called and SINCES returns control to the calling routine.

Models and Algorithms

None

Date, Programmer

December, 1976. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Set in initialization loop from 1 to the number of detectors
IB	I	Hold Area	Dummy array for use in DRWS call
J	I	Do-loop index	Set from 1 to 3 in loop over the number of incident detection algorithms
K	I	Hold Area	Set to the number of the algorithm to be implemented
NTEM	I	Hold Area	File number of the on-line detector data

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/A4/IDET
/CONTRL/IFN, IOP
/CRASH/JFIN, INCALG, INCODE, IZERO, KALL, LASTOF,
LASTON, MAXCEL, OCCOLD

5.34 Subroutine SPOINT

Program Narrative

Purpose

The purpose of SPOINT is to act upon the raw detector data and process it to obtain both detector specific and station specific occupancies.

Assumptions

None.

Approach

The routine reads a block of detector data and scans it to determine which detectors had on and off pulses. When a pair of pulses (on/off) is identified, the occupancy counters are updated and processing continues.

Synopsis

SPOINT first reads a block of raw detector data from a temporary data file. All of the actuations on that file are then scanned in sequence. For each detector actuated, the new state (detector currently on/off) is compared with the existing state. If an on/off pair of pulses can be identified, the amount of time the detector was on is computed and the occupancy counter is updated. At this point the routine loops back to read a new block of detector data.

After all detector data has been read, the occupancy counters for all detectors currently on are updated until the end of the updating interval. Percent occupancy is then computed. The occupancy at each detector station used for incident detection is then computed by aggregating the occupancy over each detector at the station.

Models and Algorithms

None.

Date, Programmer

December, 1976. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ALENV	R	Hold Area	Floating point vehicle length in feet.
DLEN	R	Hold Area	Detector loop length.
ENVN	R	Hold Area	Number of detectors at a station.
EOCC2	R	Hold Area	Scaled occupancy.
I	I	Hold Area	Counter of number of detector actuations in data block.
IBACK	I	Hold Area	Flag indicating which section of code to return to.
IBIN (1010)	I	Hold Area	Input buffer
ICIN (3,336)	I	Hold Area	Input detector data equivalenced to IBIN.
IENDET	I	Hold Area	Flag indicating the last station in the last sequence.
IOFF	I	Hold Area	Time detector turned off, milliseconds.
ION	I	Hold Are	Time detector turned on, milliseconds.
ISPRED	I	Hold Area	Elapsed time detector was on, milliseconds.
IZ	I	Do-Loop Index	Loop over previous time periods set from one to four.
J	I	Hold Area	Flag (0,1) if detector (is not, is) used for incident detection.

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	Hold Area	Counter of detector stations.
KONOFF	I	Hold Area	Detector status flag = (0,1) if detector (off, on) if doppler detector, set to speed, MPH*10.
KTEST	I	Hold Area	Flag (0,1) if detector (never, previously) actuated.
KTI	I	Hold Area	Time of actuation, milliseconds.
L	I	Hold Area	Detector type.
LEND	I	Hold Area	Detector loop length.
NIZ	I	Hold Area	Index to previous occupancy data.
NTEM	I	Hold Area	File number of detector data files.
NVDE	I	Hold Area	Detector number.
NUMBER	I	Hold Area	Number of actuations in a block of data.
NUMDET	I	Hold Area	Number of detectors at a station.
OCC	R	Hold Area	Percent occupancy at a detector.
OCC2	R	Hold Area	Percent occupancy at a detector.
VAL	R	Hold Area	Occupancy updating interval, milliseconds.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/ DTCTR,IDET

/CONTRL/ IFN,IFORM,IIDMOD,IIDOP,ILEN,IOP

/CRASH/ IFIN,INTVAL,IOCC,IOCC2,IZERO,

KALL,KON,KPOINT,LASTOF,LASTON,LENTHV,LOCATE,

OCCS

6. FUEL OVERLAY

This overlay provides link-specific evaluations of fuel consumption and vehicle emissions throughout a simulation run. This overlay uses a data tape created during the simulation activities. The structure of the tape is described in this section.

6.1 Description of FUEL Data Tape Contents

For each subinterval, 2 records:

Record Type #1 : K

Where K = number of vehicles described in record 2

Record Type #2: {[JVACC(I,J), I = 1,2], J = 1,k}

Where, for vehicle J:

JVACC (1,J) = ZZZYXXX

ZZZ = Position of vehicle in feet from
the upstream node

Y = link type = (1,2,3) for (freeway,
ramp, surface) link

XXX = speed, fps

JVACC (2,J) = +wwvuuu

ww = magnitude of acceleration or
deceleration, fpss

v = vehicle type, $1 \leq v \leq 5$

uuu = link number

sign = (+,-) if vehicle is (accelerating,
decelerating)

Records 1 and 2 are output each time the JVACC buffer
is filled during simulation and at the end of each sub-
interval.

At end of subinterval:

Record Type #3: K

Where K = -IRSCT

At end of all data:

Record Type #4: K

Where K = 0

6.2 Program FUEL

Program Narrative

Purpose

The FUEL overlay computes fuel consumption and emissions data based on vehicle trajectory data either produced on-line by a simulation run or previously stored on tape.

Assumptions

FUEL assumes, if JFUEL = 1, that the user has provided a previously generated tape containing trajectory data from a past simulation run.

Approach

The first call to FUEL is for initialization. Subsequent calls, one for each subinterval, accumulate data, print out tables, and save the data on temporary files to be accumulated over remaining subintervals.

Synopsis

The first call to FUEL is for initialization. Constants are initialized. Header records are read from or written to tape 23 if necessary. Any exogenous data cards are read in and checked diagnostically. Cumulative tables are initialized. Data tables are printed out if requested and stored on disk files if necessary. The program returns to main supervisor after setting flags for initialization complete and for next overlay to be called.

Each subsequent call to FUEL processes trajectory data. First, FUEL data tables (if necessary) and cumulative tables are spooled in from units 25 and 24 respectively. If trajectory data is being read from unit 23, then the next subinterval's data is spooled from unit 23 to 22. Each vehicle trajectory word is examined, and fuel consumption, emissions, and distance travelled are aggregated by link number and vehicle type according to vehicle speed, acceleration and type.

When all subinterval data has been processed, tables are printed out showing results both on a link-specific and network-side basis, for each vehicle type and for all types together.

Finally, cumulative tables are spooled out to unit 24 to be aggregated over the next subinterval.

If data is coming from tape 23, then the next subinterval is processed immediately. Otherwise, a flag is set to indicate the next overlay to be called by INTRAS.

Models and Algorithms

None.

Date; Programmer

February, 1976, Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
CBUF (12)	R	Hold Area	Network-wide statistics, computed from cum array
CUM (12)	R	Hold Area	Network-wide statistics, sum of link-specific statistics
FLNK (12)	R	Hold Area	Link-specific statistics, fuel consumption and emissions for each vehicle type and fro all types
I	I	(1) Do-loop Index (2) Hold Area	Generalized index Vehicle type, computed from vehicle trajectory word
IB	I	Dummy Argument for DRWS	

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ICOD	I	Switch	Data card type: 1 - HC emissions 2 - CO emissions 3 - NO emissions 4 - fuel consumption set by card input
II	I	Pointer	Beginning of data for fuel table being printed, 0 for passenger vehicle table (Table 1), 2 for commercial vehicle table (Table 2)
ILAB	I	Hold Area	Link type, table, set = Hollerith label
ILT	I	Hold Area and Do-loop Index	Link type: 1 - freeway 2 - ramp 3 - surface computed from vehicle trajec- tory word
ISPD	I	Switch	Speed index, set by card input
ISUM	I	Hold Area	Statistics for all vehicle types, computed from LBUF array
IVC	I	Do-loop Index	Statistical table no. 1 for Table 1 (passenger and all vehicles), 2 for Table 2 (commercial vehicles)
IVTYP	I	Switch	Vehicle type, set by card input
Il	I	(1) Do-loop Limit (2) Pointer	Lower limit for exogenous data: 1 - for main card, 2 - for continuation card Beginning of fuel table data

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I2	I	(1) Do-loop Limit (2) Pointer	Upper limit for exogenous data, 19 for main card, KAC for continuation card Set = I1 + 1
I3	I	Pointer	Last data item in table = (6,5) for table of (passenger and all vehicles, commercial vehicles)
I4	I	Counter	No. of individual veh-types in table, = (2,3) for table of passenger and all vehicles, commercial vehicles)
J	I	Do-loop Index	
JCO	I	Hold Area	CO emissions label, set in data statement
JDIST (3)	I	Hold Area	Network-wide fuel consumption by vehicle type
FUELT (3)	I	Hold Area	Network-wide fuel consumption by vehicle type
JHC	I	Hold Area	HC emissions label, set in data statement
JJ	I	(1) Do-loop Index (2) Hold Area	Speed at beginning of time- step (from final speed, acceleration, and time-step)
JNO	I	Hold Area	NOX emissions label, set in data statement
JSUM	I	Hold Area	Distance travelled for all vehicle types

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	(1) Do-loop Index (2) Hold Area (3) Switch	Acceleration Set = ($>0, 0, <0$) for (number of vehicle trajectories, end of all subinterval data, end of this subinterval)
KA	I	Hold Area	Acceleration index
KAC	I	Hold Area	Total number of acceleration values
KACMAX	I	Hold Area	Maximum acceleration value, set in data statement
KACMIN	I	Hold Area	Minimum acceleration value, set in data statement
KAC1	I	Hold Area	Index number of acceleration = 0, set = $-KACMIN + 1$
KDIST (110,5)	I	Hold Area	Link-specific distance tra- velled by vehicle type
KFUELT (110,5)	I	Hold Area	Link-specific fuel consumption by vehicle type
KK	I	Hold Area	Link-specific distance travelled
KS	I	Switch	Acceleration flag = (+1,-1) if vehicle is (accelerating, decelerating)
KSP	I	Hold Area	Maximum speed index, set in data statement
KSP1	I	Hold Area	Maximum allowable speed, = $KSP - 1$

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KSS	I	(1) Hold Area (2) Hold Area	Average speed, computed from speed, acceleration, time-step Fuel consumption value, from KFLCN array
KV	I	Hold Area	Speed at end of time-step
L	I	(1) Hold Area (2) Do-loop Index	Absolute link number, computed from relative link number
LBUF (110,4,5)	I	Hold Area	Link-specific cumulative statistics by vehicle type
M	I	Do-loop Index	Vehicle trajectory word index
MX	I	Hold Area	Number of freeway, ramp, or surface links
N	I	Hold Area	Total number of links, set = NTOTL or read in from fuel data tape
N1	I	Hold Area	Index of last freeway, ramp, or surface link
N2	I	Hold Area	Index of Last freeway, ramp, or surface link
SCALE	R	Hold Area	Scale factor for printout (100000 for fuel consumption, 1000 for emissions)
VBUF (2,256)	I	Hold Area	Vehicle trajectory data, read in from unit 22 by DRWS
VEML	R	Hold Area	Vehicle-miles (link-specific or network-wide distance travelled)

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/ERTRAN/ JEPAR, IER, IERCT, NPAR

/CFUEL/ JFUEL, MFUEL, NU23, IPTAB, NFLG

/SIZE/ MXLFR, MXLRA, MXLSU, NLFR, NLRA.

/A2/ ENDNDS, NTOTL

/S2/ IDTS

/CONTRL/ IBUF, ICTYPE, IFN, IFORM, ILEN,
ILSUB, IOP, ISTAT

/BFUEL/ KHCEM, KCOEM, KNOEM, KFLCN

6.3 BLOCK DATA BDCOEM

Program Narrative

Purpose

BLOCK DATA BDCOEM sets the default values for the CO emissions table KCOEM.

Assumptions

None.

Approach

None.

Synopsis

The data is loaded into array JB, a one-dimensional array which is equivalenced to KCOEM.

Models and Algorithms

None.

Date, Programmer

July, 1975, Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JB	I	Hold area	Equivalenced to KCOEM, set in data statement

Arguments Passed to Subroutine

None.

Common Block and Variables Referenced

/BFUEL/ KCOEM

6.4 BLOCK DATA BDFLCN

Program Narrative

Purpose

BLOCK DATA BDFLCN sets the default values for the fuel consumption table KFLCN .

Assumptions

None.

Approach

None.

Synopsis

The data is loaded into array JB, a one-dimensional array which is equivalenced to KFLCN .

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables:

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JB	I	Hold area	Equivalenced to KFLCN , set in data statement.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/BFUEL/ KFLCN

6.5 BLOCK DATA BDHCEM

Program Narrative

Purpose

BLOCK DATA BDHCEM sets the default values for the HC emissions table KHCEM.

Assumptions

None.

Approach

None.

Synopsis

The data is loaded into array JB, a one-dimensional array which is equivalenced to KHCEM.

Models and Algorithms

None.

Date, Programmer

July 1975, Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JB	I	Hold area	Equivalenced to KHCEM, set in data statement.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/BFUEL/ KHCEM

6.6 BLOCK DATA BDNOEM

Program Narrative

Purpose

BLOCK DATA BDNOEM sets the default values for the NOX emissions table KNOEM.

Assumptions

None.

Approach

None.

Synopsis

The data is loaded into array JB, a one-dimensional array which is equivalenced to KNOEM.

Models and Algorithms

None.

Date, Programmer

July 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
JB	I	Hold Area	Equivalenced to KNOEM, set in data statement

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/BFUEL/ KNOEM

7. INCES OVERLAY

This overlay performs all processing of detector data and generates incident detection, point processing, and MOE reports. It uses a detector data tape generated during a previous simulation.

7.1 Description of INCES Data Tape Contents

Each simulation run of SIFT may produce, as an option, a data tape containing detector actuation information. This tape is used as input to subsequent runs of the INCES module. All records are of a fixed 100-word length. There are two types of data written:

Data Type #1 - Physical records 1 through N of a sequential file. These records contain Header information.

<u>Record</u>	<u>Word Number</u>	<u>Contents</u>
1	1	Run Identification number
	2	Number of detectors on network
	3	Detector mode (0,1) for (digital, analog)
	4	Number of freeway links
	5	Number of ramp links
	6	Number of surface links
	7	Digital detector polling frequency
	8	Length of subinterval, sec
	9	Freeway simulation time-step
	10	Surface simulation time-step
	11	Contents of DTCTR (1,1)
	12	Contents of DTCTR (1,2)
	13	Contents of DTCTR (1,5)
	14	Contents of DTCTR (1,7)
	15-18	Same as 11-14 for detector #2
2-(N-1)	.	.
	.	.
	.	.
	1	Number of freeway links in this Record (maximum=48)
	2-22	Freeway link data for first link
	23-43	Freeway link data for second link
	.	.
	.	.
	.	.

<u>Record</u>	<u>Word Number</u>	<u>Contents</u>
N	1	Number of Freeway entry links
	2-L	Freeway link sequence contents of IFRANK array
	L-M	Contents of the ENDNDS array Contents of the LNKSTA array

Data Type #2 - Physical records N+1 through M in the sequential file. These records contain raw detector data

<u>Word Number</u>	<u>Contents</u>
1	Number of actuations in this record (maximum = 336)
2	Detector number
3	(0,1) if pulse is (on, off) for loop detectors; or, speed (hundredth of a mile per hour) for doppler detectors
4	Time of actuation in milliseconds
5-7	Same as 2-4 for second actuation
.	
.	
.	

7.2 Program INCES

Program Narrative

Purpose

INCES is the main routine in the INCES overlay. It reads data, performs diagnostic tests, and sequences calls to routines which perform MOE, point processing and incident detection functions.

Assumptions

The routine assumes that both card and tape input files are present. The tape file contains the network and detector data files from some previous simulation run. The card input specifies the options specified for the INCES run.

Approach

The tape file is searched for the proper run ID number and the network data is read and stored. All data is diagnostically checked. If no errors are found, the routine sequences the calls to routines which accumulate detector data and perform the incident detection, point processing and MOE estimation.

Synopsis

The routine first positions the data tape. If a list of run identification numbers has been requested, the routine reads through all files on the tape and lists the run I.D.'s present. The tape is then searched for the specified run ID number. A diagnostic message is output if it can not be found.

Once the proper file is located, the header information blocks are read and the proper arrays loaded. Data is then read from the type 65, 66, 67 and 68 cards. This data is diagnostically checked. Error messages are output if necessary. Storage elements are primed.

If no input errors have been detected, the routine initializes certain common storage elements. At this point, a sequence of CALL's is initiated to:

1. Gather detector for one occupancy updating period.
2. Compute occupancies and raw point processing data.
3. Compute Point processing statistics, if required.
4. Call the incident detection routines, if required.
5. Call the MOE estimation routines, if required.

This sequence of CALL's is continued until the end of the detector data is reached. When all the data has been processed, routines are called to produce the required output.

Control is then returned to the INTRAS overlay for the next case.

Models and Algorithms

None.

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables:

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
COUNT	R	Hold Area	Floating point. Count of vehicles during updating period.
DIST	I	Hold Area	Distance (feet) between detector stations specified for incident detection or MOE estimation
EVAL	R	Hold Area	Floating point value of occupancy updating interval (seconds)
HD	R	Hold Area	Floating point sum of all vehicle headways (milli-second)
HDWY	R	Hold Area	Mean vehicular headway (seconds)
I	I	Do-loop index and hold area	Counter for various do-loops and temporary hold areas
IADD	I	Hold Area	Sum of the number of surface, ramp, and freeway links
ICDTP	I	Hold Area	Set to the input card type expected at any point in the input stream
IDBLZR	I	Hold Area	Counter for number of null stations encountered. Two successive null stations signals the end of the station data

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Description, Initialization and Modification</u>
IDON	I	Hold Area	Sum of successive station numbers. If IDON is zero the end of the station input stream has been encountered.
IDP	I	Hold Area	Detector number, used as an array index
IDPOS	I	Hold Area	Index of location of first detector on the link
IDPP	I	Hold Area	Given IDP, IDPP is the next detector on a link
IDTP	I	Hold Area	Type of detector (loop, doppler, coupled loop)
IEND	I	Hold Area	Flag denoting the last detector station to be processed
IGOTCD	I	Hold Area	Flag used to determine which section of code to return to from a common section of code
II	I	Hold Area	Counter used in various loops
II1	I	Hold Area	Detector station number when handling input
II2	I	Hold Area	Another detector station number when handling input
IK	I	Hold Area	Set equal to detector type
IKP	I	Hold Area	Flag to specify plot detector
IN	I	Hold Area	Counter used in reading detector data blocks

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
INSTA	I	Hold Area	Detector station counter
IPACK2	I	Hold Area	Array used to transfer link information from data tape
IR	I	Hold Area	Null station flag
IRR	I	Hold Area	Null station flag of downstream station
ISAVE	I	Hold Area	Station Counter
ISTATN	I	Hold Area	Station number corresponding to a given detector
ISTION	I	Hold Area	Station number buffer array
IT	I	Hold Area	Link type
ITS	I	Hold Area	Temporary storage for sort
ITSS	I	Hold Area	Temporary storage for sort
IZY	I	Do-loop Index	Index of station numbers used in loop to test for duplicated stations set from 1 to LAZY
J	I	Do-loop Index Hold Area	Used in various loops
JEND	I	Hold Area	Flag indicating last station to be processed
JLD	I	Hold Area	Counter used in loading data
JSTA	I	Hold Area	Station number, index to LNKSTA array

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	Do-Loop Index Hold Area	Used in various loops
KIT	I	Hold Area	Station number, index to LNKSTA array
KK	I	Do-Loop Index	Implicit do-loop on read statement
KKT	I	Hold Area	Holds index of do-loop
KK1	I	Do-Loop Index	Loop over a list of station numbers to sort them from upstream location to down- stream location
KLD	I	Hold Area	Counter used in loading information from data tape
KMAX	I	Hold Area	Number of occupancy updating periods to be used in pro- ducing output
KNTA	I	Hold Area	Station number, index to LNKSTA array
KTPE	I	Hold Area	Detector type
K1	I	Do-Loop Index	Used in rest of stations from upstream to downstream
K2	I	Hold Area	Temporary storage used in loading data
L	I	Hold Area	Number of link on which is a given station
LASTDT	I	Hold Area	Last detector at a given station

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LAZY	I	Hold Area	Temporary storage used to check for duplicate stations
LI	I	Hold Area	Link on which a station resides
LII	I	Hold Area	Link on which a station resides
LINDEX	I	Hold Area	Index to LNKF array used in obtaining freeway data
LK	I	Hold Area	Counter for number of stations
LL	I	Hold Area	Link on which a station resides
LLEN	I	Hold Area	Link length (feet)
LNEXT	I	Hold Area	Receiving freeway link
LNSTA	I	Hold Area	Temporary storage used in station sort
M	I	Do-Loop Index	Loop over MOE Algorithm from 1 to 3
MM	I	Hold Area	Number of MOE Algorithms to be implemented
N	I	Do-Loop Index	Loop over number of detectors from 1 to IDET
NAG	I	Hold Area	Number of incident detection algorithms
NAXSTA	I	Hold Area	Temporary storage

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NC66 (3)	I	Hold Area	Array containing number of parameters required for each incident detection algorithm
NDIST	I	Hold Area	Average location of all detectors at a station
NEXC	I	Hold Area	Flag indicating the presence of another type 68 card on input
NL	I	Hold Area	Counter used in load information from data tape
NNL	I	Do-Loop Index	Set front to the number of freeway links
NTA	I	Hold Area	File number of data tape
NTGG	I	Hold Area	Number of type 66 cards expected on input
NUMDT	I	Hold Area	Count of the number of detectors at a given station
PDAT (4,100)	R	Hold Area	Buffer array to hold all point processing data
SAMP	R	Hold Area	Number of samples of speed and headway data for point processing
SPD	R	Hold Area	Sum of the raw speed data
SPEED	R	Hold Area	Mean speed at a detector
TEMPST (50,2)	R	Hold Area	Temporary storage array used to sort detector stations
VOL	R	Hold Area	Volume crossing a detector

Arguments Passed to Subroutine

None

Common Blocks and Variables Referenced

/A1/LNKF
/A4>IDET, DTCTR
/AMD/AMDAT
/A2/ENDNDS
/CRASH/EPARAM, IFIN, IMMOE, INCALG, INCODE,
INPOS, INTVAL, IOCC, IOCC2, IPPT, IUPDAT,
IZERO, KALL, KON, KPOINT, LASTON, LASTOF,
LENTHV, LNKSTA, LOCATE, MAXCEL, MAXSTA,
MOEALG, MPARAM, NSECTL, NUMSTN, OCCOLD,
OCCS
/CONTRL/IBUF, ICARD, ICTYPE, IDRUN, IFN, IFORM
IIDMOD, IIDOP, IINCON, IININ, ILEN, IOP,
I99LAS, MINSTS, NDT, NEXCAL
/S2/IDTF, IDTS
/ERTRAN/IEPAR, IER, IERCT, NPAR
/RANK/IFRANK, NFENT
/INCEST/INOUT
/PDATA/IRAN, ISAMP, LSTVU
/SIZE/IS, MXLFR, MXLRA, MXLSU, NLFO, NLFR,
NLRA, NLSU
/INCDMP/ICIN, IIDUMY

7.3 Subroutine INCl

Program Narrative

Purpose

INCl is a routine designed to search for freeway incidents using the "California Logic" (Ref. 1). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes three parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage.

The first two calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate.

On the third, and subsequent, call to INCl, a loop over all detector stations used for incident detection is initiated. Three parameters are computed; F1KT, the change in occupancy between successive stations, F3KIT, the percent change in occupancy over time at the downstream station, and F2KT, the percent change in occupancy between successive stations.

These parameters are compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was the "California Algorithm," as described in Payne (Ref. 1).

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AK1	R	Hold Area	Occupancy threshold
AK2	R	Hold Area	Percent occupancy change threshold
AK3	R	Hold Area	Percent change in occupancy over time threshold
F1KT	R	Hold Area	Defined as a change in occupancy between successive stations
F2KT	R	Hold Area	Percent change in occupancy between successive stations
F3KIT	R	Hold Area	Percent change in occupancy over time at the downstream station of the pair under consideration
ICODE	I	Hold Area	Incident code
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IFLAG	I	Hold Area	Flag indicating last station in last station sequence (0,1) is (not last, last) station
IHEAD	I	Array	Holds alphanumeric data for output messages
I1	I	Hold Area	Time, seconds, at the start of the interval
I2	I	Hold Area	Time, seconds at the end of the interval
JCODE	I	Hold Area	Incident code
K	I	Hold Area	Detector station number
KALGM	I	Hold Area	Algorithm number, = 1
L	I	Do-loop Index	Loop from 1 to the number of freeway links
M	I	Hold Area	Present incident code
N	I	Hold Area	Node number
NODEDN	I	Hold Area	Downstream node number
NODE 1	I	Hold Area	Upstream node number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ENDNDS

/CONTRL/IDRUN

/CRASH/EPARAM, IFIN, INCODE, INPOS, INTVAL,
IZERO, KALL, LOCATE, MAXSTA, OCCS, OCCOLD

/S2/ITIMEF

/SIZE/MXLFR

7.4 Subroutine INC2

Program Narrative

Purpose

INC2 is a routine designed to search for freeway incidents using an algorithm developed by Payne (Ref. 2). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes that six input parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage. On the first call to the routine, the incident detection frequency is tested and the run is aborted if it is found to be invalid.

The first two calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate.

On the third, and subsequent, call to INC2, a loop over all detector stations used for incident detection is initiated. Four parameters are computed; F1, the change in occupancy between successive stations, F2, the percent change in occupancy over time at the downstream station, F3, the percent change in occupancy between successive stations, and F4, the occupancy at the downstream station.

These parameters are compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was developed by Payne (Ref. 2). Specifically it is Payne's Algorithm 8. It has been implemented with a user specified number of compression wave suppression periods.

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
F1	R	Hold Area	Defined as a change in occupancy between successive stations
F2	R	Hold Area	Percent change in occupancy over time at the downstream station of the pair under consideration
F3	R	Hold Area	Percent change in occupancy between successive stations
F4	R	Hold Area	The occupancy at the downstream station of the pair under consideration
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link
IHEAD	I	Array	Holds alphanumeric data for output messages
IST	I	Hold Area	Current incident status at given pair of stations
ISUPR	I	Hold Area	The number of compression wave suppression periods input
ITV	I	Hold Area	Incident updating interval in seconds
I1	I	Hold Area	Time, seconds, at the start of the interval

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initilaization and Modification</u>
I2	I	Hold Area	Time, seconds at the end of the interval
I2T	I	Hold Area	Interval incident code
K	I	Hold Area	Detector station number
KABORT	I	Hold Area	Abort flag
KALGM	I	Hold Area	Algorithm number = 2
KIST	I	Hold Area	Flag to determine whether a tentative incident is considered an actual incident
L	I	Do-loop Index	Loop from 1 to the number of freeway links
M	I	Hold Area	Present incident code
N	I	Hold Area	Node number
NIST	I	Hold Area	Number of compression wave suppression periods plus 1
NNST	I	Do-loop index	Set from 1 to NIST
NODEDN	I	Hold Area	Downstream node number
NODE 1	I	Hold Area	Upstream node number
T1	R	Hold Area	Occupancy threshold
T2	R	Hold Area	Occupancy threshold
T3	R	Hold Area	Occupancy threshold
T4	R	Hold Area	Occupancy threshold
T5	R	Hold Area	Occupancy threshold

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ENDNDS

/CONTRL/IDRUN

/CRASH/EPARAM, IFIN, INCODE, INPOS, INTVAL,
IZERO, KALL, LOCATE, MAXSTA, OCCS,
OCCOLD

/S2/ITIMEF

/SIZE/MXLFR

7.5 Subroutine INC3

Program Narrative

Purpose

INC3 is a routine designed to search for freeway incidents using an algorithm incorporating the technique of double exponential smoothing (Ref. 3). If an incident is detected, an appropriate message is output.

Assumptions

The routine assumes that four input parameters are available in the EPARAM array.

Approach

The routine loops over all detector stations being used for incident detection and computes certain incident triggers. These triggers are then compared to threshold values obtained on input. If the threshold values are exceeded, an incident code is set. A message is output if an incident is detected.

Synopsis

All threshold values are initially retrieved from a common storage. On the first call to the routine, the incident detection frequency is tested and the run is aborted if it is found to be invalid.

Up to the first four calls to the routine are dummies with no incident detection attempted. This is required to allow a sufficient time history of occupancies to accumulate, and to initialize the time series required for the exponential smoothing.

On subsequent calls to INC3, a loop over all detector stations used for incident detection is initiated. One user specified parameter of two possible parameters is computed. The possible parameters are: FEAT(1), the change in occupancy between successive stations, and FEAT(2), the percent change

in occupancy over time at the downstream station.

The computed feature is compared to the input thresholds and an incident code is identified. This code is then loaded into the appropriate storage array.

If an incident is detected, its location is determined and an appropriate message is printed out.

Models and Algorithms

The algorithm implemented was described in (Ref. 3).

Date, Programmer

January, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Hold Area	Exponential smoothing coefficient (between 0.0 and 1.0)
ALPHA	R	Hold Area	Absolute deviate smoothing coefficient (between 0.0 and 1.0)
AT	R	Hold Area	Temporary storage for smoothing computation
BT	R	Hold Area	Temporary storage for smoothing computation
ET	R	Hold Area	Difference between the predicted value and the actual value of the required feature

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
FEAT (2)	R	Hold Array	Array holding occupancy feature being used FEAT(1) Defined as a change in occupancy between successive stations FEAT(2) Percent change in occupancy between successive stations
FEST	R	Hold Area	Predicted value of required feature
IFEAT	I	Hold Area	Requested feature (1 or 2)
IFEET1	I	Hold Area	The distance in feet of a station from the upstream node of the freeway link
IFLAG	I	Hold Area	Flag indicating the last station in the last station sequence (0,1) for (not last, last) station
IHEAD	I	Hold Area	Hold alphanumeric data for output messages
I1	I	Hold Area	Time, seconds, at the start of the interval
I2	I	Hold Area	Time, seconds, at the end of the interval
JABORT	I	Hold Area	Abort flag
JK	I	Hold Area	Incident code
K	I	Hold Area	Detector station number
KABORT	I	Hold Area	Abort flag
KALGM	I	Hold Area	Algorithm number = 3

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KJTM	I	Hold Area	Number of times routine has been called
L	I	Do-loop Index	Loop from 1 to the number of freeway links
LABORT	I	Hold Area	Abort flag
M	I	Hold Area	Present incident code
MK	I	Hold Area	Incident code in previous time-step
N	I	Hold Area	Node number
NODEDN	I	Hold Area	Downstream node number
NODE1	I	Hold Area	Upstream node number
NSUPP	I	Hold Area	Number of algorithm suppression periods (1,4) of feature (1,2) is requested
SIGMA	R	Hold Area	Temporary storage used to compute initial standard deviate
T	R	Hold Area	Incident threshold
TAU	R	Hold Area	Time parameter. Equal to 1 for a 60 second update
ZT	R	Hold Area	Computed incident trigger

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage arrays where data for this algorithm is found

Common Blocks and Variables Referenced

/A2/ENDNDS

/CONTRL/IDRUN

/CRASH/EPARAM,IFIN,INCODE,INPOS,INTVAL,
IZERO,KALL,LOCATE,MAXSTA,OCCS,
OCCOLD

/DBLEXP/SMTH2,SSQ,YERR,ABSD

/S2/ITIMEF

/SIZE/MXLFR

7.6 Subroutine MOE1

Program Narrative

Purpose

MOE1 is a routine designed to compute certain measures of effectiveness from data stored on a file during a previous simulation run. The measures computed include section density and speed.

Assumptions

The routine assumes three input parameters have been stored in the MPARAM array. The algorithm assumes that there are no exit or entrance ramps on the section under consideration.

Approach

The routine loops over all sections and computes vehicle inflows and outflows. These values, along with estimates of mean section speeds are used to estimate the value of section density. The results are stored in an output buffer.

Synopsis

On the first call to the routine, input parameters are loaded and converted to floating point numbers.

On all calls to the MOE1 routine, a subroutine is called which computes an estimate of travel time over each of the sections under consideration. A loop over all detector stations is then initiated. On the first call to the routine an initial rough count of vehicles is made using mean occupancy.

Vehicle inflows and outflows are then computed. Using this data, an estimate of section density is made. Finally, vehicle speeds, section density and section input and output volumes are stored on the output buffer.

The loop is terminated.

The final action in the routine is to update the Kalman filter in preparation for the next call.

Models and Algorithms

The algorithm used in routine MOE1 is based on the work of Gazis and Knapp (Ref. 4). Density is estimated based upon initial conditions, vehicle inflows and outflows. Kalman Filtering techniques are used.

Date, Programmer

March, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ALANE	R	Hold Area	Number of lanes in section
ALEN	R	Hold Area	Vehicle length in feet
B	R	Hold Area	Initial Kalman filter between 0.0 and 1.0
BNEW	R	Hold Area	Temporary storage for density computation
C(2)	R	Hold Area	Section input and output Vehicles per updating period
CC	R	Hold Area	Temporary storage for Kalman filter update
CNT	R	Hold Area	Number of vehicles crossing a detector

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
D	R	Hold Area	Kalman filter constant
DD	R	Hold Area	Temporary storage for Kalman filter update
DELC	R	Hold Area	Difference between vehicle inflow and outflow
DIST	R	Hold Area	Section length in feet
IENDET	I	Hold Area	Flag (0,1) for (not last, last) station in last sequence
K	I	Hold Area	Station counter
KK	I	Hold Area	Downstream station number
KKK	I	Hold Area	Station pointer
KSTA	I	Hold Area	Station number
L	I	Do-loop Index	Loop from 1 to 2 to compute inflows and outflows
LINDEX	I	Hold Area	Index to LNKF array
LINK	I	Hold Area	Link number
ND	I	Hold Area	Detector number
Q	R	Hold Area	Variance of trap error term (veh**2)
R	R	Hold Area	Rough count error variance (veh**2)
RD	R	Hold Area	Temporary storage for Kalman filter update
TVAL	R	Hold Area	MOE updating interval

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
YNEW	R	Hold Area	Current count of vehicles in a given section
YOLD (100)	R	Hold Area	Count of vehicles in a section in previous update period
ZKR	R	Hold Area	Number of vehicles entering a section which cannot leave in current update interval
ZOLD (100)	R	Hold Area	Number of vehicles in the section during last update period which could not exit during updating period

Argument Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage where data for this algorithm is found

Common Blocks and Variables Referenced

/A1/NLV

/A4/DTCTR

/AMD/AMDAT

/CRASH/KALL, KPOINT, LENTHV, LNKSTA, LOCATE,
MPARAM, NSECTL

/MTOC/TOCC

/PDATA/ISAMP, IRAW

/SIZE/NLFO

/TRVTM/TT

7.7 Subroutine MOE2

Program Narrative

Purpose

MOE2 is a routine designed to compute certain measures of effectiveness from data stored on a file during a previous simulation run. The measures computed include section density and speed.

Assumptions

The routine assumes three input parameters have been stored in the MPARAM array. The algorithm assumes that there are no exit or entrance ramps on the section under consideration.

Approach

The routine loops over all sections and computes vehicle inflows and outflows. These values, along with estimates of mean section speeds are used to estimate the value of section density. The results are stored in an output buffer.

Synopsis

On the first call to the routine, input parameters are loaded and converted to floating point numbers.

On all calls to the MOE2 routine, a subroutine is called which computes an estimate of travel time over each of the sections under consideration. A loop over all detector stations is then initiated. On the first call to the routine an initial rough count of vehicles is made using mean occupancy.

Vehicle inflows and outflows are then computed. Using this data, an estimate of section density is made. Finally, vehicle speeds, section density and section input and output volumes are stored on the output buffer. The loop is terminated.

The final action in the routine is to update the Kalman filter in preparation for the next call.

Models and Algorithms

The algorithm used in routine MOE2 is based on the work of Gazis and Szeto (Ref. 5). Density is estimated based upon initial conditions, vehicle inflows and outflows. Kalman Filtering techniques are used.

Date, Programmer

March, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ALANE	R	Hold Area	Number of lanes in section
ALEN	R	Hold Area	Vehicle length in feet
C(2)	R	Hold Area	Section input and output Vehicles per updating period
CNT	R	Hold Area	Number of vehicles crossing a detector
DELC	R	Hold Area	Difference between vehicle inflow and outflow
DIST	R	Hold Area	Section length in feet
G	R	Hold Area	Kalman filter coefficient
IENDET	I	Hold Area	Flag (0,1) for (not last, last) station in last sequence
K	I	Hold Area	Station counter

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KK	I	Hold Area	Downstream station number
KKK	I	Hold Area	Station pointer
KSTA	I	Hold Area	Station number
L	I	Do-loop Index	Loop from 1 to 2 to compute inflows and outflows
LINDEX	I	Hold Area	Index to LNKF array
LINK	I	Hold Area	Link number
ND	I	Hold Area	Detector number
Q	R	Hold Area	Variance of error term (veh**2)
RHO	R	Hold Area	Ratio of System/Observation Noise between 0.0 and 1.0
SIGMA	R	Hold Area	Initial count estimation vari- ance error term (veh**2)
SIGMA1	R	Hold Area	Temporary storage for Kalman filter update
SIGMA2	R	Hold Area	Temporary storage for Kalman filter update
TVAL	R	Hold Area	MOE update interval
XK (100)	R	Hold Area	Running count of the number of vehicles in a section
XN	R	Hold Area	Number of vehicles in a section during previous time-step
XNEW	R	Hold Area	Current estimate of number of vehicles in a section
Z	R	Hold Area	Initial estimate of number of vehicles in a section

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LOCP	I	Hold Area	Input, location in common storage where data for this algorithm is found

Common Blocks and Variables Referenced

/A1/NLV
 /A4/DTCTR
 /AMD/AMDAT
 /CRASH/KALL,KPOINT,LENTHV,LNKSTA,LOCATE,
 MPARAM,NSECTL
 /MTOC/TOCC
 /PDATA/ISAMP,IRAW
 /SIZE/NLFO
 /TRVTM/TT

7.8 Subroutine MOE3

Program Narrative

Purpose

MOE3 is a routine designed to compute certain measures of effectiveness from data stored on a file during a previous simulation run. The measures computed include section density and speed.

Assumptions

The routine assumes three input parameters have been stored in the MPARAM array. The algorithm assumes that there are no exit or entrance ramps on the section under consideration.

Approach

The routine loops over all sections and computes vehicle inflows and outflows. These values, along with estimates of mean section speeds are used to estimate the value of section density. The results are stored in an output buffer.

Synopsis

On the first call to the routine, input parameters are loaded and converted to floating point numbers. Parameters are checked for validity.

On all calls to the MOE3 routine, a subroutine is called which computes an estimate of travel time over each of the sections under consideration. A loop over all detector stations is then initiated. On the first call to the routine an initial rough count of vehicles is made using mean occupancy.

Vehicle inflows and outflows are then computed. Using this data, an estimate of section density is made. Finally, vehicle speeds, section density and section input and output volumes are stored on the output buffer. The loop is terminated.

The final action in the routine is to update the Kalman filter in preparation for the next call.

Models and Algorithms

The algorithm is based upon the work by Nahi and Trivedi (Ref. 6). It is a recursive estimator of section density and average speed. Density is estimated based upon initial conditions, vehicle inflows and outflows. Kalman Filtering techniques are used.

Date, Programmer

March, 1977. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ALANE	R	Hold Area	Number of lanes in section
ALEN	R	Hold Area	Vehicle length in feet
C (2)	R	Hold Area	Section input and output Vehicle per updating period
DIST	R	Hold Area	Section length in feet
EL	R	Hold Area	Number of lane-miles in the section
F(100)	R	Hold Area	Station specific Kalman filter
FPSMPH	R	Hold Area	Conversion from feet per second to miles per hour
IENDET	I	Hold Area	Flag (0,1) for (not last, last) station in last sequence
K	I	Hold Area	Station counter

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KK	I	Hold Area	Downstream station number
KKK	I	Hold Area	Station pointer
KSTA	I	Hold Area	Station number
L	I	Do-loop Index	Loop from 1 to 2 to compute inflows and outflows
LINDEX	I	Hold Area	Index to LNKF array
LINK	I	Hold Area	Link number
ND	I	Hold Area	Detector number
P	R	Hold Area	Section density error variance term
PO	R	Hold Area	Initial expected value of the section density error variance (veh/lane-mi)**2
RAT	R	Hold Area	Ratio of system noise variance to observation noise variance
REST	R	Hold Area	Computed section density (veh/lane-mi)
RHO (100)	R	Hold Area	Previous value of section density
S	R	Hold Area	Updating interval in hours
SIGAM2	R	Hold Area	Observation noise variance
SIGPH2	R	Hold Area	System noise variance
TVAL	R	Hold Area	Updating period
U(100)	R	Hold Area	Section specific speed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>

X	R	Hold Area	Section length in miles
XRAW	R	Hold Area	Detector specific vehicle count

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>

LOCP	I	Hold Area	Input, location in common storage where data for this algorithm is found

Common Blocks and Variables Referenced

/A1/NLV
 /A4/DTCTR
 /AMD/AMDAT
 /CRASH/KALL,KPOINT,LENTHV,LNKSTA,LOCATE,
 MPARAM,NSECTL
 /MTOC/TOCC
 /PDATA/ISAMP,IRAW
 /SIZE/NLFO
 /TRVTM/TT

7.9 Subroutine MOUT

Program Narrative

Purpose

MOUT reads MOE data from a temporary data file and formats it. The results are the standard MOE output.

Assumptions

The routine assumes that the temporary data file has been written properly during the course of the INCES run.

Approach

The routine loops over all algorithms and all time-steps to format the MOE output.

Synopsis

The routine loops over all MOE algorithms. For each algorithm, the output is formatted into groups of three MOE sections. Each section is defined by its upstream and downstream detector stations. The MOE data for each section is read off the data file for every time interval. Output buffers are loaded and dumped. The routine then loops back to get data for a new time interval. After all time intervals have been handled, the routine chooses a new set of three stations. The data file is rewound and the process begins anew.

Models and Algorithms

None.

Date, Programmer

January 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AMDAT (3,4,100)	R	Hold Area	MOE input buffer
IEND	I	Hold Area	Code to denote last detector station in the last sequence
II	I	Hold Area	Counter for number of sections considered on this pass (up to three)
IJ	I	Do-loop Index	Loop over four pieces of MOE data
IJK	I	Do-loop Index	Loop over sections considered on this pass. Set from one to a maximum of three
IJKL	I	Hold Area	Output buffer index
IK (3)	I	Hold Area	Array holding section numbers for up to three sections considered
IKK	I	Hold Area	Section number
INOUT (300)	I	Hold Area	Dummy input buffer
IST (3,2)	I	Hold Area	Array holding station numbers for up to three sections
JJ	I	Do-loop Index	Index of initialization loop. Set from one to two.
JLK (3,2)	I	Hold Area	Array containing link numbers for sections being considered
JTM	I	Hold Area	MOE updating time, seconds
K	I	Hold Area	Counter for station number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KC	I	Hold Area	Counter for number of data file reads
KSTA1	I	Hold Area	Upstream station number
KSTA2	I	Hold Area	Downstream station number
LL	I	Do-loop Index	Index of implied do-loops in write statements
M	I	Do-loop Index	Index of number of MOE algorithms implemented. Set from one to three
N	I	Hold Area	Temporary storage for packed station data
QUT (12)	R	Hold Area	Output buffer
PDAT (400)	R	Hold Area	Dummy input buffer

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KMAX	I	Hold Area	Input, maximum number of time intervals for which data is available

Common Blocks and Variables Referenced

/CONTRL/ IDRUN, IFN, IOP

/CRASH/ IUPDAT, LNKSTA, LOCATE, MOEALG

7.10 Subroutine PINC

Program Narrative

Purpose

PINC reads incident detection data from a temporary data file and formats it. The results are the standard incident detection output.

Assumptions

The routine assumes the temporary data file has been written properly during the course of the INCES run.

Approach

The routine loops over all algorithms and time-steps to format the appropriate report.

Synopsis

The incident detection output is formatted into groups of six incident detection traps. Each trap is defined by its upstream and downstream detector station number. The routine identifies the six traps to be handled. The data file is rewound. It is read for each of the updating intervals. The incident code is read for each trap and the proper alphanumeric identifier is stored in an output buffer. The routine prints one line of alphanumeric codes and returns to get data for a new updating interval. After the last updating interval has been handled, a new group of six traps is identified, the data file is rewound, and the process begins anew.

Models and Algorithms

None.

Date, Programmer

January 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AMDAT (1200)	R	Hold Area	Dummy input buffer
IEND	I	Hold Area	Code to denote last detector station in the last sequence
II	I	Hold Area	Counter for number of traps considered on this pass (up to six)
IJK	I	Do-loop Index	Loop over stations considered on this pass. Set from one to a maximum of six
IK(6)	I	Hold Area	Array holding trap numbers for up to six traps considered
IKK	I	Hold Area	Trap number
INOUT (3,100)	I	Hold Area	Incident code input buffer
IOUT(6)	I	Hold Area	Output array holding alpha-numeric data
IST(6,2)	I	Hold Area	Array holding station numbers for up to six traps
JI	I	Do-loop Index	Index in initialization loop. Set from one to two
JLK(6,2)	I	Hold Area	Array containing link numbers for traps being considered
JOUT(6)	I	Hold Area	Output buffer
JTM	I	Hold Area	Evaluation time, seconds
K	I	Hold Area	Counter for station numbers

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KC	I	Hold Area	Counter for number of data file reads
KKK	I	Hold Area	Incident code
KSTA1	I	Hold Area	Upstream station number
KSTA2	I	Hold Area	Downstream station number
LL	I	Do-loop Index	Index of implied do-loop in write statements
M	I	Do-loop Index	Loop over all possible detec- tion algorithms set from one to three
N	I	Hold Area	Temporary storage for packed station data
NNP	I	Hold Area	Number of parameters used for a given algorithm
PDAT (400)	R	Hold Area	Dummy input buffer

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KMAX	I	Hold Area	Input, maximum number of time intervals for which data is available

Common Blocks and Variables Referenced

/CONTRL/ IDRUN, IFN, IOP

/CRASH/ EPARAM, INCALG, INTVAL, LNKSTA,
LOCATE, MAXCEL

/NUMP/ NPARS

7.11 Subroutine POINT

Program Narrative

Purpose

The purpose of POINT is to act upon the raw detector data and process it to obtain both detector specific and station specific occupancies. In addition, raw data for the point processing statistics are accumulated.

Assumptions

POINT assumes that the correct detector data is available on the appropriate peripheral file.

Approach

The routine reads a block of detector data and scans it to determine which detectors had on and off pulses. When a pair of pulses (on/off) is identified, the occupancy counters and point proceeding statistics are updated and processing continues.

Synopsis

POINT first reads a block of raw detector data from a temporary data file. All of the actuations on that file are then scanned in sequence. For each detector actuated, the new state (detector currently on/off) is compared with the existing state. If an on/off pair of pulses can be identified, the amount of time the detector was on is computed and the occupancy counter is updated.

In addition to the occupancy accumulation, the routine gathers statistics on individual vehicle speeds across detectors, headways and detector specific vehicle counts. Vehicle speeds are gathered for doppler and single loop detectors directly. Coupled loop detectors are handled by predicting a range of arrival times at the downstream detector of the pair. Speeds are computed when the next arrival at the downstream detector occurs within the predicted range of arrival times.

After all detector data has been read, the occupancy counters for all detectors currently on are updated until

the end of the updating interval. Percent occupancy is then computed. The occupancy at each detector station used for incident detection is then computed by aggregating the occupancy over each detector at the station.

Models and Algorithms

None.

Date, Programmer

December, 1976. Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ALENV	R	Hold Area	Floating point vehicle length in feet
DIST	R	Hold Area	Distance between coupled loop detectors
DLEN	R	Hold Area	Detector loop length
ENUM	R	Hold Area	Number of detectors at a station
EOCC2	R	Hold Area	Scaled occupancy
I	I	Hold Area	Counter of number of detector actuations in data block
IBACK	I	Hold Area	Flag indicating which section of code to return to
IBIN (1010)	I	Hold Area	Input buffer
ICIN (3,336)	I	Hold Area	Input detector data equiva- lenced to IBIN

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IOPD	I	Hold Area	Flag (0,>10) if detector is (doppler, plot)
IDTP	I	Hold Area	Detector type
IENDET	I	Hold Area	Flag indicating the last sta- tion in the last sequence
IOFF	I	Hold Area	Time detector turned off, milliseconds
ION	I	Hold Area	Time detector turned on, milliseconds
ISPRED	I	Hold Area	Elapsed time detector was on, milliseconds
ITMAX	I	Hold Area	Latest time vehicle may arrive at downstream detector of coupled pair
ITMIN	I	Hold Area	Earliest time vehicle may arrive at downstream detector of coupled loop
IT4	I	Hold Area	Integer estimate of time of arrival at downstream detector of coupled loop
IVU	I	Hold Area	Vehicle speed estimate (fps)
IZ	I	Do-loop Index	Loop over previous time periods set from one to four
J	I	Hold Area	Flag (0,1) if detector (is not, is) used for incident detection
K	I	Hold Area	Counter of detector stations
KONOFF	I	Hold Area	Detector status flag = (0,1) if detector (off,on) if doppler detector, set to speed, MPH*10

7.12 Subroutine POUT

Program Narrative

Purpose

POUT reads point processing data from a temporary data file and formats it. The result is the standard point processing output.

Assumptions

The routine assumes that the temporary data file has been written properly during the course of the INCES run.

Approach

The routine processes data and formats the appropriate output report.

Synopsis

The routine logic loops over all detectors. For each detector the data file is rewound and up to 18 updating periods worth of data are read. Output buffer arrays are primed. Once the arrays are primed, the buffer arrays are printed. The routine then goes back to read a new set of updating periods.

Models and Algorithms

None.

Date, Programmer

January 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AMDAT (3,4,100)	R	Hold Area	Dummy input buffer
I	I	Do-loop Index	Set from 1 to the number of detectors

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IDN	I	Hold Area	Downstream node number of the link on which a given detector is located
IDTP	I	Hold Area	Detector type (0,1,3,4) for (doppler, short loop, downstream, upstream part of coupled pair)
II	I	Do-loop Index	Set from 1 to the number of updating intervals to be output (up to 18)
INOUT (3,100)	I	Hold Area	Dummy input buffer
IOUT (4,3)	I	Hold Area	Storage array for alphanumeric output
IUP	I	Hold Area	Upstream node number of the link on which a given detector is located
J	I	Do-loop Index	Loop to transfer data from input buffer. Set from 1 to 4
JNTV	I	Hold Area	Point processing updating interval, seconds
K	I	Hold Area	Counter of number of updating intervals processed
KOUT(18)	I	Hold Area	Volume output buffer
K1	I	Hold Area	Index relating to type of output (1,2,3,4) for (volume, speed, headway, occupancy)
K10	I	Hold Area	Lower bound of evaluation time for output loop
K20	I	Hold Area	Upper bound of evaluation time for output loop

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
L	I	Hold Area	Link number on which detector is located
LANE	I	Hold Area	Lane in which detector is located
LINDEX	I	Hold Area	Index to LNKF array
LL	I	Do-loop Index	Loop over all freeway links
LLL	I	Do-loop Index	Implied do-loop in write statement
NUDE	I	Hold Area	Detector number
OUT (4,18)	R	Hold Area	Output buffer
PDAT (4,100)	R	Hold Area	Point processing input buffer

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KMAX	I	Hold Area	Input, set to the number of evaluation periods for which data is available

Common Blocks and Variables Referenced

/A1/ NLV

/A2/ ENDNDS

/A4/ DTCTR, IDET

/CONTRL/ IDRUN, IFN, IOP

/CRASH/ IUPDAT

/SIZE/ MXLFR, NLFO

7.13 Subroutine READET

Program Narrative

Purpose

The purpose of this routine is to transfer detector data from the input tape to a temporary data file.

Assumptions

None.

Approach

The input tape is searched for actuations which fall in the specified time block. The data is unpacked and transferred to the temporary file.

Synopsis

Both the input data tape and the temporary data file are repositioned. The input file is then searched for all actuations occurring between time IZERO and IFIN. As actuations are identified, they are unpacked and loaded into the temporary file sequentially.

After all data has been transferred, the temporary file is closed with an end file. Should an end-of-file be encountered on the input tape, an error message is generated.

Models and Algorithms

None.

Date, Programmer

February 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Loop from 1 to the number of elements in a data block
IBIN (1010)	I	Hold Area	Input buffer array
ICIN (2,500)	I	Hold Area	Input array equivalenced to IBIN
JBAR	I	Hold Area	Counter of number of actuations encountered
JBOUT (1010)	I	Hold Area	Output buffer arrays
JCIN (3,336)	I	Hold Area	Output array equivalenced to JBOUT
KTI	I	Hold Area	Time of detector actuation
NTA	I	Hold Area	Input tape file number
NTEM	I	Hold Area	Output tape file number
NUMBER	I	Hold Area	Number of elements in the input buffer

Arguments passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
IEND	I	Switch	Output, Code (0,1,2) if end of detector input data (not encountered, encountered with data, encountered without data). Set in subroutine

Common Blocks and Variables Referenced

/CONTRL/IFN,IFORM,ILEN,IOP
/CRASH/IFIN,IZERO
/ERTRAN/IEPAR,IER,IERCT,NPAR

7.14 Subroutine TTIME

Program Narrative

Purpose

TTIME computes an estimate of the travel time between the upstream and downstream detector stations of a freeway section.

Assumptions

No doppler radar detectors may be present.

Approach

Travel time is defined as the inverse of the speed at the upstream and downstream detector stations.

Synopsis

The routine loops over all freeway sections. The mean speed over the upstream detector station is computed and stored. The same is done at the downstream station. The loop is terminated when mean speeds at all the relevant stations are computed. The routine then loops over all freeway sections. This time, the section mean speeds are computed by averaging the mean speeds at the upstream and downstream stations. This speed is converted to travel time after the section length is determined.

Models and Algorithms

None.

Date, Programmer

February 1977. Reuben Goldblatt.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AN	R	Hold Area	Size of speed sample
DIST	R	Hold Area	Length of freeway section, feet
ENUM	R	Hold Area	Number of detectors at a station
IENDET	I	Hold Area	Flag used to denote last station in last sequence
ITYPE	I	Hold Area	Type of detector
K	I	Hold Area	Counter for number for free-way sections
KK	I	Hold Area	Downstream station number
NUDE	I	Hold Area	Detector number
NUMDET	I	Hold Area	Number of detectors at a given station
RAW	R	Hold Area	Sum of the speeds at a detector station
V(100)	R	Hold Area	Array of mean speeds at a detector station
VV	R	Hold Area	Sum of mean speeds at upstream and downstream station

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/ DTCTR

/CRASH/ KPOINT, LOCATE, NSECTL

/PDATA/ IRAW, ISAMP

/TRVTM/ TT

8. POSPRO OVERLAY

This overlay creates a file of simulation statistics on the Statistical Data Tape for future processing by the SAM Module.

8.1 Program POSPRO

Program Narrative

Purpose

Stores results of a simulation run on tape for future analysis.

Assumptions

None.

Approach

Reads results of a simulation run from a temporary disc file and copies it onto a Statistical Data Tape. MOEs are calculated on a subinterval specific as well as a network-wide basis.

Synopsis

The POSPRO module (POSt PROcessor) is designed to act as an interface between the SIFT simulation and the statistical testing procedures of the SAM module. The task performed by POSPRO is to create a file of simulation statistics on the Statistical Data Tape. The module, consisting of a single routine, is entered at the end of each simulation subinterval to add the subinterval link--specific and network statistics to a temporary file. When the total simulation run is completed, the contents of this file are added to the Statistical Data Tape. Subroutines DRWS and UNPACK are called to assist with input and output manipulations of the data so it can be transferred to the data tape for future analysis.

Models and Algorithms

None.

Date, Programmer

January 1976. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Loops over link types
IAVCON	I	Hold Area	Average content (veh) $\times$ 100
IB (880)	I	Hold Area	Parameter in argument list for DRWS routine. Contains array of data
IDN	I	Hold Area	Downstream node number
IHOLD (5)	I	Hold Area	Holds cumulative statistics
IIIIII	I	Hold Area	Density for freeway in (vehicles per lane mile) $\times$ 100
IM	I	Do-loop Index	Set from 1 to N. Loops over subintervals
IUP	I	Hold Area	Upstream node number
IZ (8,110)	I	Hold Area	Array used to store all MOE values
KLD	I	Switch	(1,2) for (freeway, non- freeway)
K1 K3 K4 K5 K6 K7 K8	I	Hold Area	Used as index in IPACK array
LAREA	I	Hold Area	Total link area

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LAREAF	I	Hold Area	Total area of freeway links
LAST	I	Hold Area	Last file flag
LF	I	Hold Area	Number of freeway links
LR	I	Hold Area	Number of ramp links
LS	I	Hold Area	Number of surface links
L1	I	Hold Area	Index for IVFEET array
N	I	Hold Area	Number of subintervals
NET (5,2)	I	Hold Area	Contains network wide MOEs
NETF (6)	I	Hold Area	Network data for freeways
NFILE1	I	Constant	Temporary storage file for statistical output
NFILE2	I	Constant	Temporary storage file for cumulative statistics
NFILE3	I	Constant	Output data tape
NUM	I	Hold Area	Total content of freeway links
SUBTIM (26)	I	Hold Area	Stores length, in time, of each subinterval

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A2/ ENDNDS, IVFEET

/A8/ CLEN, IPACK

/CONTRL/ IFN, IFORM, ILEN, IOP,
 IRSCT, ISTAT, ISTFLG, NDT,
 NEXCAL

/SIZE/ MXLFR, MXLRA, MXLSV, NLFR,
 NLRA

9. INPLOT OVERLAY

The purpose of INPLOT is to create vehicle trajectory and MOE contour plots using standard CALCOMP plotting calls. A data tape, created during simulation, is used.

9.1 Description of Plot Data Tape Contents

Each simulation run of SIFT may produce, as an option, a binary sequential file of raw data. This data will consist of five record types each of which conveys some portion of the raw data required for plotting.

Record Type #1 - Physical record 1 of sequential file.

<u>Word Number</u>	<u>Contents</u>
1	Simulation Run Identification
2	Number of links, n , of contour plot raw data included in this file, and contour plot time step, Δt_1 , in minutes
3	Longitudinal spacing of MOE stations (feet)
4	Number of links and lanes, m , of trajectory plot raw data included in this file, and trajectory plot time step, Δt_2 , in seconds

Record Type #2 - Physical record 2 of sequential file.

<u>Word Number</u>	<u>Contents</u>
1	Link identification(upstream and downstream node numbers)
2	Length of link
3	Longitudinal position of first MOE station (feet)
4,5,6 . . .	Same as Words 1,2 and 3 for second link of contour information
$3_{n-2}, 3_{n-1}, 3_n$	Same as Words 1,2 and 3 for nth link of contour information

Record Type #3 - Physical record 3 for sequential file.

<u>Word number</u>	<u>Contents</u>
1	XXXYYYZW, where: XXX = upstream node YYY = downstream node Z = lane W = receiving lane
2	Length of link
3,4	Same as words 1 and 2 for second link-lane of vehicle trajectory information
.	
.	
.	
$2_{n-1}, 2_n$	Same as words 1 and 2 for nth link-lane of vehicle trajectory information
2_{n+1}	flag = 2 indicating new record follows

Record Type #4 - Physical record 4 and beyond of sequential file. At each time interval indicated by Δt_1 , a series of n of these records will occur in the sequential file. The ith record of this series will contain the following information.

<u>Word Number</u>	<u>Contents</u>
1	Spot speed (mph) at first longitudinal station on link i.
2	Volume (veh/hr) at first longitudinal station on link i.
3	Speed at second longitudinal station on link i.
4	Volume at second longitudinal station on link i.
.	
.	
.	
$2_{k-1}, 2_k$	Speed and volume at kth (final) longitudinal station on link i.

Word NumberContents 2_{k+1}

A flag word indicating the next record type (this is essentially a validity check)

= 0 if last record of case

= 1 if continuation record follows

= 2 if new record follows

Record Type #5 - physical record 5 and beyond of sequential file. At each time interval dictated by Δt_2 , a series of at least m of these records will occur in the sequential file. The record(s) corresponding to the i th link-lane set of trajectory data will contain the following information:

Word NumberContents

1

Vehicle identification for first (upstream) vehicle in link-lane i

2

Longitudinal position of first vehicle

3

Vehicle ID for second vehicle

4

Longitudinal position of second vehicle

.

.

.

.

99,100

Identification and position of 50th vehicle (if it exists)

101

A flag word indicating the next record type (for validity checking) and, whether there is a continuation record for link-lane i caused by greater than 50 vehicles.

= 0 if last record

= 1 if continuation record follows

= 2 if new record follows

The order of occurrence of record types 4 and 5 in the sequential file is predictable based upon Δt_1 , Δt_2 , n and m .

When both types of records are produced at the same simulation time step, the type 5 records will occur first. The flag word for the last record of a run will indicate termination of data. The first type 5 record is at time = 0; the first type 4 record is at time Δt_1 .

Record Type #6 - End of all data cases on tape.

<u>Word Number</u>	<u>Contents</u>
1	-999999
2	-999999

9.2 Program INPLOT

Program Narrative

Purpose

The INPLOT overlay is the supervisor program for all plot-related operations.

Assumptions

INPLOT assumes that the user has provided a plot data tape containing appropriate trajectory and/or contour data generated by a past simulation run; the tape must be defined as unit 19. Also, if actual plots are to be produced, a CALCOMP output tape must be provided, defined as unit 12 and having the attributes required by the CALCOMP plotter.

Approach

As each type 70 card is read, the appropriate action is taken, either to read contour change parameters, get a plot tape file index, or produce a plot. When all type 70 cards have been read, control is returned to the INTRAS supervisor.

Synopsis

Each type 70 card is read and checked diagnostically. If a tape file index is requested, FILEX is called, and then the next type 70 card, if any, is read.

If the request is for a contour change, the revised contour parameters are read from the card. INPLOT then assumes that another type 70 card follows.

If a plot is requested, various parameters for plot size, time and distance scales, etc. must be taken from the type 70 card. Then the case data requested by case ID must be found on tape unit 19 and spooled to unit 20 and 21. The first header record is read from unit 20.

If a contour plot was requested, CONTR is called to produce plots for one or all MOEs, depending on the user request.

If a trajectory plot was requested, the next header record must be read from unit 20. Then, either TRAJEC is called (if a plot for one lane was requested), or else SPTAL is called (if all lanes were requested).

If more type 70 cards follow, they are processed in the same way. When all cards have been processed, control returns to the INTRAS supervisor.

If at any time an error condition occurs, the current plot is aborted and processing skips to the next case.

Models and Algorithms

None.

Date, Programmer

July 1976. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
AFLG	I	Switch	Origin of call to TRAJEC, set = 0 to indicate TRAJEC called from INPLOT
I	I	Do-loop Index	(1) Number of contour values; (2) Number of link-lanes
IB	I	Dummy Argument for DRWS	
IDSCAL	I	Hold Area	Distance scale (ft/inch), read in on input card
IFLG	I	Hold Area	End-of-record flag, read from trajectory data file

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ITSCAL	I	Hold Area	Time scale, tenths-of-a-second, read in on input card
ITYPE	I	Hold Area	Card type, read in on input card
J	I	Do-loop Index	LNKID index, set = (1,2) for word (1,2)
JJ	I	Counter	Trajectory record buffer index, initialized to 0; incremented from 1 to m * 2
LFLAG	I	Switch	Last plot request flag, read in on input card, set = (0,1) for (no more, more) plot requests
MD	I	Hold Area	XXXYY, XXX = Number of trajec- tory link-lanes YY = Time between data points, sec. Read in from trajectory file
ND	I	Hold Area	XXXYY, XXX = Number of contour links YY = Time between data points, min. Read in from trajectory file
OLDRUN	I	Hold Area	Previous case run ID requested, initialized to 0, set = RUNID after case found
SIMID	I	Hold Area	ID of case or data tape, read in from trajectory file
XX	R	Hold Area	Revised contour value, from input card

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, ICN, IFORM, ILEN, IOP, IPL, NEXCAL

/ERTRAN/ IEPAR, IER, IERIT, NPAR

/IPERR/ IPERR

/CRD/ IMOLE, IREQ, LANE, RUNID, XORG, XPAGE, YORG,
YPAGE

/TMDAT/ DELT1, DELT2, DSCALE, KPLOT, NXT1, NXT2,
TIME1, TIME2, TSCALE

/T1/ MAXV, MAXT

/PLT/ DNODE, JBUF, LNKID, M, N, UNODE

/CNTR/ SGAP, CTR

9.3 Subroutine AXPLOT

Program Narrative

Purpose

AXPLOT sets up the page, and draws and labels the x and y axes.

Assumptions

None.

Approach

First, the page outline is set up. Then the y-axis and x-axis are drawn.

Synopsis

First, all character heights needed by PAGE, LEGEND, and AXPLOT are defined. PAGE is called to set up the page outline, plot heading and roadway sections. If a contour plot was requested, LEGEND is called to plot the key to contour symbols. Then the y-axis, hash-marks and accompanying scale are drawn (the y-axis represents time for contour plots, distance for trajectory plots). Finally, the x-axis, hash-marks and scale are drawn (the x-axis represents distance for contour plots, time for trajectory plots).

Models and Algorithms

None.

Data, Programmer

July, 1975, Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A	R	Hold Area	Absolute y-coordinate (inches)
ClA	R	Hold Area	6/7 of hash-mark label height (inches)
ClB	R	Hold Area	4/7 of hash-mark label height (inches)
C2A	R	Hold Area	6/7 of axis label height (inches)
FI	R	Hold Area	Absolute hash-mark location (inches)
I	I	General Purpose Do-loop Index	
IDEC	I	Switch	Number of decimal places in hash-mark label, set = -1 for distance scale (no decimal point), set = +1 for time scale (one decimal place)
IW	I	Hold Area	Time scale $\times 100$
IX	I	Hold Area	Number of hash-marks on x-axis
IY	I	Hold Area	Number of hash-marks on y-axis
ND	I	Hold Area	Node number in node sequence to be plotted
T	R	Hold Area	Time or distance value at hash-mark (min. or feet)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
X	R	Hold Area	Absolute x-coordinate of lower left corner of y-axis hash- mark label
XI	R	Hold Area	Absolute hash-mark location
XTIME1	R	Hold Area	Plot start time (mins.)
XX	R	Hold Area	Absolute x-coordinate of lower left corner of hash-mark label on x-axis
X1	R	Hold Area	Absolute x-coordinate of start of x-axis label
X2	R	Hold Area	Absolute x-coordinate of start of y-axis label
Y	R	Hold Area	Absolute y-coordinate of end of y-axis
YY	R	Hold Area	Absolute y-coordinate of lower left corner of x-axis hash- mark label
Y1	R	Hold Area	Absolute y-coordinate of start of x-axis label
Y2	R	Hold Area	Absolute y-coordinate of start of y-axis label

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/PLT/ LEN, LKCNT, LNKID, NDLIST
/TMDAT/ DSCALE, KPLOT, TIME1, TIME2,
TSCALE, START
/IPERR/ IPERR
/CRD/ DISPX, DISPY, IREQ, XLIM, YLIM

9.4 Subroutine COMPCT

Program Narrative

Purpose

COMPCT compacts the VPOS and IOVEH arrays w.r.t. time and vehicles so that more data may then be accumulated and plotted.

Assumptions

COMPCT assumes that VPOS and IOVEH are, in fact, full and that the data has already been plotted.

Approach

The arrays are compacted first w.r.t. time and then with respect to vehicles.

Synopsis

First, the VPOS and IOVEH arrays are compacted with respect to time; i.e., the last two time points become the first two points, and the data at previous times is discarded. Then, the data at these two time points is compacted w.r.t. vehicles; i.e., any intervening empty slots between vehicles are removed. Finally, all empty slots in the IOVEH array are flagged with 0, and all in VPOS are flagged with -1.

Models and Algorithms

None.

Date, Programmer

July, 1975, Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Vehicle index no., set in do- loops, from 1 to MAXV
J	I	Hold-area Do-loop Index	Next-to-last time point index; time point index, set in do- loop from 3 to MAXT
K	I	Pointer	Next slot to be compacted

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/T1/ MAXV, MAXT
/PLT/ IDVEH, VPOS
/TMDAT/ ISTEP.

9.5 Subroutine CONTR

Program Narrative

Purpose

CONTR processes data to create a contour plot for a given MOE (Measure Of Effectiveness).

Assumptions

CONTR assumes that the simulation contour data has been written to unit 21 by IOPROC.

Approach

Data is processed until the MOE buffer is full or all data has been processed. Then the buffer data is plotted and reinitialized, and then CONTR goes on to process any remaining data.

Synopsis

First, the contour header record is read from unit 21 and the LNKID and FSTAT arrays are filled. PATH is called to determine the node sequence to be plotted. The number of stations on each link is computed and stored in FSTAT. AXPLOTT is called to set up the page and axes. Then CONTR processes data at each time point for all links in path: the desired MOE is computed for all stations on each link and stored in the MOE array. If the array becomes filled, then CPLOT is called to plot the data, and the array is then reinitialized. CONTR then goes on to process data at the next time point. When all data has been processed, CPLOT is called to process any remaining data.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
FSTAT (100,2)	I	Hold area	FSTAT(I,1) contains first station loc. on link I FSTAT(I,2) contains no. stations on link I
I	I	General do-loop index	
IB	I	Dummy argu- ment to DRWS	
JBUF (72)	I	Hold area	Buffer into which contour header record is read, read in by DRWS
POS	I	Hold area	Number of stations upstream of first station on link being processed
POSPJ	I	Hold area	Index of station being processed
REC (101)	I	Hold area	Buffer into which contour data records are read, read in by DRWS
XR1	R	Hold area	Spot speed - mph
XR2	R	Hold area	Volume - veh/hr/ln

Arguments Passed To Subroutine

None.

Common Blocks and Variables Referenced

/CNTR/ FS, SGAD, SMAX
/TMDAT/ DELT1, START, TIME1, TIME2
/CRD/ IMOE, LANE, RUNID
/PLT/ LEN, LKCNT, LNKID, MOE, N, NOLIST
/IPERR/ IPERR
/T1/ MAXV, MAXT
/GRTRAN/ IER, IEPAR, IERCT,NPAR
/CONTRL/ IFN, IFORM, ILEN, IOP

9.6 Subroutine CPLOT

Program Narrative

Purpose

CPLOT computes and plots data for all stations and time points in the MOE buffer.

Assumptions

None.

Approach

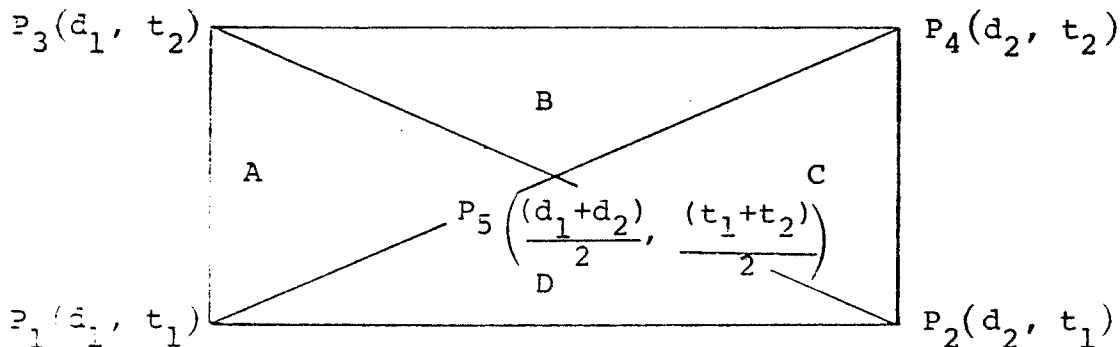
CPLOT loops through all stations and time points, computes the contours, and then plots them.

Synopsis

At each data point for station I, time step I, CPLOT looks at the set of four points containing (1) this point; (2) station J, time step I + 1; (3) J + 1, I, (4) J + 1, I + 1. It then computes the contours by interpolation and plots them. Descriptive symbols, explained in the key drawn by LEGEND, are placed along the contours.

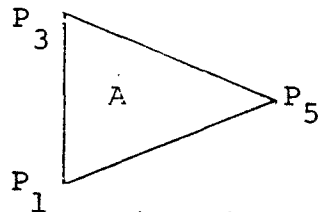
Models and Algorithms

The contours are defined as follows: consider the four data points for station J, time I; J + 1, I; J, I + 1; J + 1, I + 1. A fifth point, midway in time and distance is considered, defining four triangles.



With each P_i is associated an MOE value C_i .

Then each of the four triangles (A,B,C,D) is processed to determine the contour passing through it. Take triangle A as an example:



First, if all vertices have the same MOE value, then they define a contour which will be plotted if they correspond to a contour in the CTR array.

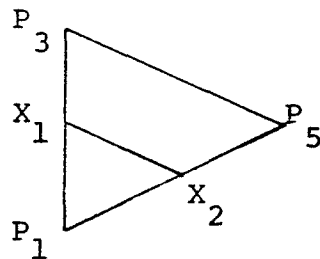
If they are different, then they are ordered by increasing MOE value. Suppose

$$C_1 < C_3 < C_5 .$$

Then, by table look-up, the contour value in the CTR array, K, is found such that

$$C_1 \leq K \leq C_5 .$$

By linear interpolation, we can get the points X₁, X₂ which represent contour K



A line is drawn from X₁ to X₂. The same procedure is followed for the other three triangles.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
C(5,3)	R	Hold area	Abs. coordinates and MOE values at 5 points from which contours will be determined
DIFF	R	Hold area	Ratio of (difference between contour and low MOE value)/ (difference between high and low MOE values)
I	I	Do-loop Index	Time step index, set in do-loop, from 1, IS
IH	I	Hold area	Index of highest MOE value in triangle
IM	I	Hold area	Index of median MOE value in triangle
IL	I	Hold area	Index of lowest MOE value in triangle
IS	I	Hold area	No. time-steps -1
IT	I	Hold area	Temp. storage of triangle vertex when reordering
J	I	Do-loop Index	Station index, set in do-loop from 1, MS
JSYM	I	Hold area	Index of special character representing contour being plotted
K	I	Do-loop Index	Contour index, set in do-loop from 1, 14
L	I	Do-loop Index	Triangle index, set in do-loop from 1, 4
LL	I	Hold area	Points to next triangle vertex

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
MS	I	Hold area	No. Stations -1
X	R	Hold area	Absolute X-coordinate of first point of contour segment
XGAP	R	Hold area	Distance between stations
X1	R	Hold area	Absolute X-coordinate of second point of contour segment
Y	R	Hold area	Absolute Y-coordinate of first point of contour segment
Y1	R	Hold area	Absolute Y-coordinate of second point of contour segment
YY	R	Hold area	Data plotting offset (= data time interval/(2 x time scale)

Arguments Passed To Subroutine

None.

Common Blocks and Variables Referenced

/CNTR/ CTR, FS, SGAP, SMAX
 /CRD/ DISPX, DISPY, IMOE
 /TMDAT/ DELT1, DSCALE, ISTEP, START,
 TSCALE
 /PLT/ MOE

9.7 Subroutine FILEX

Program Narrative

Purpose

FILEX prints a tape file index for the simulation plot data tape.

Assumptions

FILEX assumes the user has provided a plot data tape created by a previous simulation run.

Approach

FILEX loops through all cases on the tape and prints the link information for each case.

Synopsis

First, IOPROC is called to read the data for the next case. The first header record is read from unit 20. Next, all contour link information, if any, is read from unit 21 and printed out, 50 lines at a time. Finally, all trajectory link-lane information is printed out, 50 lines at a time.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Line number being printed, set in do-loop from N1, N2
IB	I	Dummy argu- ment to DRWS	
IDWN	I	Hold area	Downstream node of link being processed
IR	I	Hold area	Temporary storage while unpack- ing trajectory link information
IREC	I	Hold area	Receiving lane
IUP	I	Hold area	Upstream node of link being processed
J	I	Do-loop Index	No. links or link lanes, set in do-loop from 1, N or 1, M
JJ	I	Counter	Next word to be copied from one buffer to another
K	I	Do-loop Index	No. words per link, set in do- loop from 1,3 or 1,2
LANE	I	Hold area	Lane number
MD	I	Hold area	Word 4 from header record
ND	I	Hold area	Word 2 from header record
NPAGE	I	Counter	Page number, incremented by 1 each time a page is printed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>

N1	I	Hold area	Index of link printed on first line of page

N2	I	Hold area	Index of link printed on last line of page

Arguments Passed To Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IFN, IFORM, ILEN

/IPERR/ IPERR

/PLT/ IBUF2 (equivalenced to VBUF),
 IBUF3 (equivalenced to VBUF),
 JBUF (equivalenced to IDVEH),
 M,N

9.8 Subroutine IOPROC

Program Narrative

Purpose

IOPROC reads the simulation plot data tape and copies data for one case to the appropriate disk files.

Assumptions

IOPROC assumes the user has provided a plot data tape created by a previous simulation run.

Approach

The next record type is determined and the next record is read and copied to the appropriate disk file, until all records for this case have been read.

Synopsis

The next record is read from tape 19 and checked for end-of-all-data-cases flag. If so, then IOPROC returns with IPERR = 1, meaning no more cases were found. If not, then the tape is backspaced and the record is reread as the header record type 1.

The number of contour links and trajectory link-lanes and time steps are taken from the header record. It is then copied to unit 20.

If there is contour data on tape, then the contour header record type 2 is read and copied to unit 21.

If there is trajectory data on tape, then the trajectory header record type 3 is read and copied to unit 20.

If there is trajectory data and it is time to read it, then records type 5 are read, one for each link-lane plus continuation records, if any. They are then copied to unit 20.

If there is contour data and it is time to read it, records type 4 are read, one for each link, plus continuation records, if any.

When the last record for this case has been read, IOPROC returns to the calling program.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Link or link-lanes, set in do-loop number from 1,N or 1,M
IGAP	I	Hold area	Gap between stations
IRECT	I	Hold area	Last record type read
I1	I	Hold area	Index of link length for link I
I2	I	Hold area	Index of first station for link I
JJ	I	Do-loop Index	Contour record number for link I, set in do-loop from 1, KK
K	I	Hold area	Record length
KK	I	Hold area	Number contour records for link I
NUMST	I	Hold area	Number stations on link I

Arguments Passed To Subroutine

None.

Common Blocks and Variables Referenced

/PLT/ BUFR (equivalenced to VPOS),
 BUF2 (equivalenced to IDVEH), LNKID,
 N,M,

/TMDAT/ DELT1, DELT2, NXT1, NXT2

/IPERR/ IPERR

/CONTRL/ IFN, IFORM, ILEN, IOP

9.9 Subroutine LEGEND

Program Narrative

Purpose

LEGEND draws a legend adjoining each contour plot giving a key to the contour symbols and corresponding values.

Assumptions

None.

Approach

LEGEND draws the outline, the title, and then the symbols and meanings.

Synopsis

First, an outline is drawn to the right of the plot, three inches wide. Then the heading is drawn. Finally, for each of the 14 contour values, the symbol, units and value are drawn in table format.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
C7	R	Hold area	Character height
C7A	R	Hold area	Character width

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop index	Contour index, set in do-loop from 1, 14
IDEC	I	Hold area	No. of decimal places in contour value (=-1 if no decimal point)
X	R	Hold area	Absolute x-coord. of lower left corner of legend
SC	R	Hold area	Absolute x-coord. of center of contour symbol
XD	R	Hold area	Abs. x-coord. of start of contour value
XE	R	Hold area	Absolute x-coord. of start of contour units
XX	R	Hold area	Absolute x-coord. of start of heading
Y	R	Hold area	Absolute y-coord. of line being drawn
YC	R	Hold area	Absolute y-coord. of center of contour symbol
Z	R	Hold area	Contour value

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CNTR/ CTR

/CRD/ DISPY, IMOE, XORG, XPAGE, YORG,
YPAGE

9.10 Subroutine PAGE

Program Narrative

Purpose

PAGE draws page outline, writes heading and roadway section indicators, and determines axes lengths and locations.

Assumptions

None.

Approach

PAGE performs its functions appropriately for either a contour or trajectory plot.

Synopsis

If this is not the first plot, then the pen is first moved to a new location on the paper. The page outline is then drawn. The margins, axis lengths and displacements are determined to allow space for the roadway section indicators (on the left side for trajectory plots, and on the bottom for contour plots). Then the user-requested start and stop times are checked to assure adequate axis lengths. Next, the appropriate plot heading is drawn. Then the roadway section guide, indicating node numbers and link lengths drawn to scale, is drawn -- below the x-axis for contour plots, to the left of the y-axis for trajectory plots. As each node is processed, its location is checked for being out of the distance range specified by the user. Dotted lines are drawn perpendicular to the roadway section guide at each node to indicate the data for each link.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
C1A	R	Hold area	Width of hash mark labels
C3A	R	Hold area	Width of top heading
C4A	R	Hold area	Width of roadway section label
C6A	R	Hold area	Width of node number within roadway section box
DIG	R	Hold area	No. digits in node number
DIST(20)	R	Hold area	Coordinates along distance axis where link boundary dotted lines will be drawn
I	I	General do- loop index	
IPEN	I	Switch	Pen position = (2,3) for (down, up)
IXAX	I	Hold area	X-axis length
IYAX	I	Hold area	Y-axis length
J	I	Hold area	Upstream node
K	I	Hold area	Downstream node
ND	I	Hold area	Index in NDLIST of node being processed
XAX	R	Hold area	X-axis length
XINC	R	Hold area	Distance between successive x-coordinates when drawing dotted lines for link boundaries

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
XK	R	Hold area	Node number
XLEFT	R	Hold area	Total width of left margin labels
XLEN	R	Hold area	Link length
XM	R	Hold area	Absolute location of right end of x-axis
XMARG	R	Hold area	Width of left and right margins
XR	R	Hold area	Node number or lane number
XRT	R	Hold area	Total width of right margin labels, if any
XTOT	R	Hold area	Total width of left and right labels (XLEFT + XRT)
XX	R	Hold area	General absolute x-coordinate
X1	R	Hold area	General absolute x-coordinate
X2	R	Hold area	General absolute x-coordinate
X3	R	Hold area	General absolute x-coordinate
YAX	R	Hold area	Y-axis length
YBTM	R	Hold area	Width of total bottom labels
YINC	R	Hold area	Distance between successive y-coordinates when drawing link boundary dotted lines
YM	R	Hold area	Absolute location of upper end of y-axis
YMARG	R	Hold area	Width of top and bottom margins

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
YTOP	R	Hold area	Width of total top labels
YTOT	R	Hold area	Total width of top and bottom labels (YTOP + YBTM)
YY	R	Hold area	General absolute y-coordinate
Y1	R	Hold area	General absolute y-coordinate
Y2	R	Hold area	General absolute y-coordinate
Y3	R	Hold area	General absolute y-coordinate
ZLIM	R	Hold area	Time axis length

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/LANET/ LANET
/TMDAT/ DSCALE, TIME1, TIME2, TSCALE
/PLT/ DNODE, LKCNT, LNKID, NDLIST, UNODE
/CRD/ DISPX, DISPY, IMOE, IREQ, LANE,
RUNID, XLIM, XORG, XPAGE, YLIM,
YORG, YPAGE
/CHT/ CL, C2, C3, C4, C5, C6
/ERTRAN/ IER, NPAR, IEPAR
/CONTRL/ IPL
/IPERR/ IPERR

9.11 Subroutine PATH

Program Narrative

Purpose

PATH searches all link data to find a node sequence defining a path from the upstream to downstream node specified by user.

Assumptions

None.

Approach

All paths are examined; if the desired one is not found, then the plot is aborted.

Synopsis

The LNKID array is searched for the first link with the desired upstream node. If a lane must be matched, then that, too, is checked. Then the downstream node is checked; if it matches the desired downstream node, then the path has been found. If not, then the link having this node as its upstream node is found, and the procedure is repeated. As each candidate link is found, its index is saved in the NDLIST array. If trajectory plots for all paths are requested, then the upstream and downstream nodes are saved in the NDS array. If a link is eliminated from consideration, it is flagged as such so that it will not be checked again.

If the end of a path has been reached without finding a downstream node match, then the last candidate link found is thrown out and the link data is searched for another path. This backing-up procedure is continued until the downstream node has been matched, or all possible paths have been exhausted. If the latter occurs, an error message is printed, and the plot is aborted.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Link index. Set in do-loop from II, MN
II	I	Hold area	First candidate link index
IREC	I	Hold area	Receiving lane
K	I	Counter	Number of NDLIST entries
KK	I	Hold area	Previous NDLIST entry
LN	I	Hold area	Lane number
LNKOUT (100)	I	Switch	LNKOUT(I) = (0,1) if link I (has not, has) been examined
MN	I	Hold area	Maximum no. links or link-lanes
NODE	I	Hold area	Upstream node to be matched
N1	I	Hold area	Upstream node of candidate link
N2	I	Hold area	Downstream node of candidate link

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/PLT/ DNODE, LKCN, LNKID, VPOS, M, N,
NDS, UNODE

/ERTRN/ IEPAR, IER, NPAR

/IPERR/ IPERR

/CRD/ IREQ, LANE, RUNID

9.12 Subroutine SEARCH

Program Narrative

Purpose

SEARCH searches the vehicle ID list to find the index of a desired vehicle or, if none exists, to find an empty slot for it.

Assumptions

SEARCH assumes the input parameter L was set to 0 by the calling program.

Approach

SEARCH loops through the entire vehicle list until the desired slot is found.

Synopsis

SEARCH loops through each entry in the vehicle ID list. If a match is found with the desired ID, then L is set to the index of that entry and SEARCH returns to the calling program; otherwise, the next slot is examined. If the vehicle is not found among the present ID's, then it is assigned the first empty slot in the list. If no match and no empty slot are found, then SEARCH returns to the calling program without resetting L.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop Index	Vehicle ID index. Set in do-loop from 1, MAXV

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
L	I	Hold area	Vehicle Index. Input: set = 0 by calling subroutine. Output: set = vehicle index if slot found
VID	I	Hold area	Input, vehicle ID for which slot must be found. Set by calling program

Common Blocks and Variables Referenced

/PLT/ IDVEH
/TI/ MAXV

9.13 Subroutine SPTAL

Program Narrative

Purpose

SPTAL is called if trajectory plots for all lanes are requested. It determines all the paths to be plotted and then calls TRAJEC to plot each path.

Assumptions

None.

Approach

First all the paths are determined according to receiving lanes. Then a plot is drawn for each path.

Synopsis

First PATH is called to determine the node sequence for the desired path. Then the LANES array is set up, which contains the sequence of receiving lanes for each link in the path. Next, for each lane, the sequence of LNKID indices must be determined and stored in the NDLIST array. TRAJEC is then called to draw a plot for this lane. As each link-lane is processed, its entry in the LANES array is flagged as processed. When all paths have been processed, control returns to the calling program.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Link-lane index, link-lane index in path to be plotted
IB	I	Dummy argu- ment to DRWS	
II	I	Do-loop Index	Link-lane index
J	I	Do-loop Index	Link-lane index, lane index in path to be plotted
JBUF(48)	I	Hold area	Link data buffer
JJ	I	Do-loop Index	Link data word index
K	I	Do-loop Index	Link-lane index
KFLG	I	Switch	TRAJEC flag, set = 1 to indicate TRAJEC being called from SPTAL
KK	I	Counter	Word number of JBUF word being stored in LNKID array
LANES (20,9)	I	Hold area	LANES (I,J) contains receiving lane (or -1 if no receiving lane) of lane J, link I, where I is index of NDS array
LINK	I	Hold area	Link node and lane data to be matched with LNKID array
LN	I	Hold area	Lane number

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>

NLNK	I	Hold area	Maximum number link-lanes in path to be plotted

NXLN	I	Hold area	Receiving lane

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IFN, IFORM, ILEN, IOP

/IPERR/ IPERR

/LANET/ LANET

/PLT/ LNKID, M, NDLIST, NDS

9.14 Subroutine TPLOT

Program Narrative

Purpose

TPLOT plots the time-distance trajectories for all vehicles in the VPOS buffer.

Assumptions

None.

Approach

TPLOT loops through all vehicles at all time-points, skipping those for which there is no data.

Synopsis

TPLOT loops through all vehicles. For each vehicle, it checks for data at all time points. If this is the first time point, or if there is not data at previous time points, the pen is up. If it is not a continuous trajectory, the pen is also up. Otherwise, the pen is down. The pen is then moved to the data point, whose coordinates have been converted to inches. When all data points have been processed, a new start time is computed to reflect that the data buffer will be reinitialized.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Vehicle index no., from 1, MAXV
IPEN	I	Switch	IPEN = (2,3) for position = (down, up)
J	I	Do-loop Index	Time point index no. from 1, MAXT
T	R	Hold area	Absolute x-coordinate
V	R	Hold area	Absolute y-coordinate
XV	R	Hold area	Vehicle position

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/TMDAT/ DELT2, DSCALE, NXT2, START, TSCALE
/CRD/ DISPX, DISPY
/PLT/ IDVEH, V
/T1/ MAXV, MAXT

9.15 Subroutine TRAJEC --

Program Narrative

Purpose

TRAJEC generates one trajectory plot for a given path.

Assumptions

None.

Approach

Data is stored in the VPOS buffer, and the buffer is plotted and reinitialized whenever it becomes full.

Synopsis

First, PATH is called if only one lane has been requested. Then AXPLOT is called to set up the page and axes. Then, at each time point, all trajectory data records are examined. If the time is not within the user-specified range, then all records are skipped. If within the time range, it must be stored in the VPOS array. A new or existing slot must be found by SEARCH for each vehicle in the record, and the vehicle position is stored. If no position is found, COMPCT is called to compact the buffer, and SEARCH is called again. If the buffer is still full, the plot must be aborted.

When all points in the user-specified time interval have been plotted, or when the buffer is full, TPLOT is called to plot the data. If necessary, COMPCT is called to compact the buffer, and then any remaining data is similarly processed.

Models and Algorithms

None.

Date, Programmer

July, 1975. Nancy Rosenfield.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
I	I	Do-loop Index	Link-lane index, from 1, M
K	I	Do-loop Index	Link-lane index of path to be plotted; trajectory word index
L	I	Do-Loop Index and Hold area	Vehicle index in IDVEH and VPOS arrays
REC(101)	I	Hold area	Buffer holding trajectory records read by DRWS.

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization, and Modification</u>
KFLG	I	Switch	Input, KFLG = 0 if TRAJEC called from INPLOT (only one lane plotted), 1 if called from SPTAL (all lanes plotted)

Common Blocks and Variables Referenced

/TMDAT/ DELT2, ISTEP, NXT2, START, TIME1,
TIME2, TSCALE
/PLT/ IDVEH, LKCN, LNKID, M, NDLIST, VPOS
/ERTRAN/ IEPAR, IER, NPAR
/CONTRL/ IFN, IFORM, ILEN, IOP
/IPERR/ IPERR
/TI/ MAXV, MAXT
/CRD/ RUNID

10. SAM OVERLAY

The Statistical Analysis Module performs a number of statistical tests on the data tape created by POSPRO. Comparisons can be made between the results of two simulation runs, or between the results of a simulation run and a field data collection study.

10.1 Description of Statistical Data Tape Contents

At the end of each simulation subinterval, the subinterval link-specific and network statistics are recorded. All such statistics are then copied to the Statistical Data Tape in the following format:

Record Type #1 - Physical record 1 of sequential file.

<u>Word Number</u>	<u>Contents</u>
1	Run identification number
2	Number of subintervals, N, in run
3	Number of freeway links, LF, in network
4	Number of ramp links, LR, in network
5	Number of surface links, LS, in network

Record Type #2 - Physical record 2 of sequential file.

<u>Word Number</u>	<u>Contents</u>
1	Duration of first subinterval (minutes $\times$ 100)
2	Same as word 1 for second subinterval (if applicable)
.	
.	
.	
N	Same as word 1 for the N th subinterval

Record Type #3 - Physical record 3 of sequential file.

<u>Word Number</u>	<u>Contents</u>
1	XXXXYY where: XXX = upstream node number of freeway link 1 YYY = downstream node number of freeway link 1
2	Same as word 1 for freeway link 2
.	
.	
.	
LF	Same as word 1 for freeway link LF

Record Type #4 - Physical record 4 of sequential file.
Same as record type 3 for ramp links.
This record should contain LR words.

Record Type #5 - Physical record 5 of sequential file.
Same as record type 3 for surface links.
This record should contain LS words.

Record type #6 through #10 occur for each subinterval. Physical records 6 through 10 are record type #6 through #10 for the first subinterval. Physical records 11 through 15 are record type #6 through #10 for the second subinterval and so forth. Therefore, there should be 5N+5 MOE data records written.

Record Type #6 - Contains MOE data for each freeway link.

<u>Word Number</u>	<u>Contents</u>
1	Number of vehicles discharged from freeway link 1
2	Vehicle minutes $\times 100$
3	Delay time (minutes) $\times 100$
4	Number of lane changes

<u>Word Number</u>	<u>Contents</u>
5	Vehicle miles $\times$ 100
6	Density (vehicles/lane-mile) $\times$ 100
7	Average speed (miles/hour) $\times$ 100
8	Volume (vehicles/lane-hour) $\times$ 100
9 through 16	Same as words 1 through 8 but for freeway link 2
.	
.	
.	
8(LF-1)+1 through 8*LF	Same as words 1 through 8 but for freeway link LF
<u>Record Type #7</u> - Contains MOE data for each ramp link.	

<u>Word Number</u>	<u>Contents</u>
1	Number of vehicles discharged from ramp link 1
2	Vehicle minutes $\times$ 100
3	Delay time (minutes) $\times$ 100
4	Stopped delay (minutes) $\times$ 100
5	Vehicle miles $\times$ 100
6	Average saturation percent $\times$ 100
7	Average speed (miles/hour) $\times$ 100
8	Not used for ramp links
9 through 16	Same as words 1 through 8 but for ramp link 2
.	
.	
.	
8(LR-1)+1 through 8*LR	Same as words 1 through 8 but for ramp link LR

Record Type #8 - Contains same MOE data, as described for ramp links, for each surface link in network. Length of record should be 8*LS words.

Record Type #9 - Contains MOE data for network averaged over freeway links.

<u>Word Number</u>	<u>Contents</u>
1	Number of vehicle discharged from network
2	Vehicle minutes $\times 100$
3	Delay time (minutes) $\times 100$
4	Vehicle miles $\times 100$
5	Average speed (miles/hour) $\times 100$
6	Density (vehicles/lane-mile) $\times 100$

Record Type #10 - Contains MOE data for network averaged over all links. Length of record is five words. The descriptions of each word are the same as the first five words of record type #9.

Record Type #11 - Physical record 5N+6 of sequential file. Contains network wide MOE data averaged over freeway links for all subintervals. Length and description is the same as for record type #9.

Record Type #12 - Physical record 5N+7 of sequential file. Contains network wide MOE data averaged over all links for all subintervals. Length and description is the same as for record type #10.

Record Type #13 - End of all data cases on tape.

<u>Word Number</u>	<u>Contents</u>
1	999

10.2 Program SAM

Program Narrative

Purpose

The SAM (Statistical Analysis Module) routine is used to perform statistical comparisons between pairs of simulation runs or to compare simulation results with externally supplied MOEs (i.e., field data). It can also be used for file management operations.

Assumptions

None.

Approach

This main program of the SAM module reads the SAM data cards, performs file creation or management activities, and calls subroutines to accomplish a statistical comparison between files.

Synopsis

The routine is actuated by card types 90 to 97. The cards define the activities the Statistical Module is to perform. These activities may be divided into two types of functions which the Statistical Module is to perform. They are:

- A statistical analysis comparing two networks. The MOEs to be compared can be the results of simulation runs or external MOEs from field data. Card types 92-95 are used to show the results of a field test on data tape.
- File management operation(s).

If statistical analysis is requested, the routine scans the Statistical Data tape for the appropriate files. It then calls READCL to read the data and perform calculations necessary to prime the analysis matrices. Next, PRINT is called to perform certain other basic calculations and print the data. This is followed by a call to STAT which in turn calls the appropriate routine to perform the various statistical tests and print the results.

Reports are generated by the SAM module containing comparisons and statistical tests of MOE values. The MOEs to be studied and their applicability are:

<u>MOE</u>	<u>Application</u>
Vehicles Discharged	All Links and Network
Delay Time (veh-min)	All Links and Network
Lane Changes	Freeway Links
Denisty	Freeway Links and Network
Avg. Saturation Percent	Non-Freeway Links
Vehicle-Miles	All Links and Network
Travel Time (veh-min)	All Links and Network
Volume	Freeway Links
Time in Queue (veh-min)	Non-Freeway Links
Average Speed	All Links and Network

There are various SAM report formats (tables) which are summaries of the statistical analysis and can be used for comparisons of two networks.

If the file management task is selected, the routine reads through the indicated Data Tape(s), one file at a time. Then, according to specifications in the control card type 90, it does one or more of the following in accordance with code IM. This code identifies the particular file management function requested.

IM=1 Copy contents of Statistical Data Tape A onto Statistical Data Tape B deleting certain files (which must be specified in card type 91).

IM=2 Add contents of Statistical Data Tape A to contents of Statistical Data Tape B.

IM=3 Produce a listing of the "basic" contents of Statistical Data Tape A.

IM=5 Add contents of a file on PUNCH cards. (Card types 95-97) to Statistical Data Tape A.

IM=6 Add contents of a file on PUNCH cards to empty Statistical Data Tape A.

IO= (0,1) if (only freeway links, all links) are to be included in statistical analysis (specify only if IC=1).

A number of error messages pertaining to improper input may be printed. Subroutine EROT is called to assist in this function.

Models and Algorithms

None.

Date, Programmer

December 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IADD	I	Hold Area	Sum for overall subinterval network density
IAVON	I	Hold Area	Average content for particular link in particular subinterval, veh $\times$ 100

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IB (880)	I	Hold Area	Parameter in argument list for DRWS routine. Contains array of data
ICD	I	Hold Area	Card type number
IDelet (25)	I	Hold Area	Array that stores file numbers which in turn will be deleted
IFILE	I	Switch	IFILE=Network A for I=1, Network B for I=2
II	I	General Purpose Do-loop Index	
III	I	Do-loop Index	Used as index for subinterval specific network-wide data holder
IO	I	Switch	(0,1) if (only freeway links, all links)
IREC	I	Hold Area	Number of particular record on tape
ISUB	I	Do-loop Index	Used to loop over subintervals
IZ (8,110)	I	Hold Area	Array used to hold all MOE values
K	I	Do-loop Index	Loop over ramp and then surface links
KOUNT	I	Hold Area	Counts points to next file to be deleted
LAREA	I	Hold Area	Link area in lane feed

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
LAREAF	I	Hold Area	Sum of areas over freeway links for density calculations
LAST	I	Hold Area	Last file
LFH	I	Hold Area	Number of freeway links
LRH	I	Hold Area	Number of ramp links
LT	I	Do-loop Index	Loops over link type
LUP	I	Hold Area	Points to last record on file
NET (5,2)	I	Hold Area	• Sum of freeway MOE and non-freeway MOE • Average speed for non-freeway • Average network statistics over link for each subinterval • Network MOE for subinterval
NETA	I	Constant	I.D. number of Network A
NETB	I	Constant	I.D. number of Network B
NETF (6)	I	Hold Area	Average overall density for freeway link
NETN (5,2)	I	Hold Area	Holds network-wide MOE statistics
NHOLD	I	Constant	Number of subintervals in simulation
NL	I	General Purpose	Holds number of links for each link type
NSUM	I	Hold Area	Number of subintervals

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NUM	I	Hold Area	Sum of average contents for network
NUP	I	General Purpose	Holds number of links for each link type
SUBTIM (26)	I	Hold Area	Subinterval times

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, IFORM, ILEN, IOP,
NEXCAL

/ERTRAN/ IEPAR, IER, IERCT, NPAR

/SAMP2/ ISKIP, LF, LIN, LR, LS, N,
NFILEB, NLSA, NLSA2, NT

/SAMP3/ DMAT, DMBT, DNTA, DNTB,
DONETA, DONETB

10.3 Subroutine ANOVA

Program Narrative

Purpose

Perform one-way analysis of variance test on a given pair of arrays.

Assumptions

- Samples of each array are independent and random.
- The corresponding random variables have normal distributions with the means μ_i and the common variance σ^2 .
- The error terms, e_i , are values assumed by independent random variables having normal distributions with 0 means and the common variance σ^2 .

Approach

The routine calculates the F statistic and compares it with the F-table imbedded within the routine. It then determines its significance.

Synopsis

The routine follows the standard arithmetic operations to calculate the F-statistic. It then computes the degrees of freedom which are used as a pointer to the F-table. The level of significance is determined and that information, as well as the sum of squares within, sum of squares between, and the F-statistic are stored in the argument list.

Models and Algorithms

Described in Crow, E., et al, "Statistic Manual," Dover Publications, Inc., New York, 1960, Chapter 5, Pages 118-127.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Intiialization and Modification</u>
ABAR	R	Hold Area	Mean of the A array
BBAR	R	Hold Area	Mean of the B array
EPS	R	Constant	Equals .00001. N is compared against EPS for default values
F	R	Hold Area	The F-statistic
H	R	Hold Area	Points to determine value in F-table
I	I	Do-loop Index	Set from 1 to N representing each value in the array
IDF	I	Hold Area	Degrees of freedom. Used as a pointer in the F-table
S	R	Hold Area	Points to determine value in F-table
SSB	R	Hold Area	Sum of squares ~~between the two~~ arrays
SSW	R	Hold Area	Sum of squares within the two arrays
SUMA	R	Hold Area	Sum of A array
SUMB	R	Hold Area	Sum of B array
SUMSQ	R	Hold Area	Sum of squares of the two arrays

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
TABLE (26,2)	R	Hold Area	Imbedded table of critical values of F in the F-test
TBAR	R	Hold Area	Mean of the total of two arrays
VB	R	Hold Area	Equal to SSB
VW	R	Hold Area	Equal to SSW divided by the total sample minus 2
XDF	R	Hold Area	Degrees of freedom
XN	R	Hold Area	Length of each array
XNN	R	Hold Area	XN multiplied by 2 or length of combining both arrays

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(1)	R	Hold Area	Input, array of XN values. Output, sum of squares within, sum of squares between, F-statistic
B(1)	R	Hold Area	Input, second array of XN values
ISIG	I	Switch	Output, (0,1,2) for level of significance indicator
NINT	I	Hold Area	Output, length of each array

Common Blocks and Variables Referenced

None.

10.4 Subroutine CALC

Program Narrative

Purpose

This routine calculates for each of these arrays, the arithmetic mean and variance.

Assumptions

Samples of each array are random.

Approach

The routine performs the appropriate arithmetic operations to calculate the means and variance of each given array.

Synopsis

The arrays' addresses and lengths are provided by the calling routine via the argument list of the calling statement. For each array, an arithmetic mean and a variance are calculated. The means and variances are passed to the calling routine as output via the argument list of the calling statement.

Models and Algorithms

$$\text{Arithmetic mean} = \frac{\sum_{j=1}^N X_j}{N} \quad \text{where } N \text{ is number of data points in array}$$

$$\text{Variance} = \frac{\sum_{j=1}^N (X_j - \bar{X})^2}{N-1} \quad \text{where } \bar{X} \text{ is the arithmetic mean and } N \text{ is number of data points in the array}$$

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
EPS	R	Constant	Equals .00001. N is compared to EPS for default value
I	I	Do-loop Index	Set from 1 to N representing each value in the array
SUMA	R	Hold Area	Sum of all values in A array
SUMAA	R	Hold Area	Used to calculate variance of A array
SUMB	R	Hold Area	Sum of all values in B array
SUMBB	R	Hold Area	Used to calculate variance of B array
XN	R	Hold Area	Length of each array

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(2)	R	Hold Area	Input, array of length XN
ABAR	R	Hold Area	Output, mean of A array
B(2)	R	Hold Area	Input, second array of length XN
BBAR	R	Hold Area	Output, mean of B array

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NINT	I	Hold Area	Input, length of each array (number of subintervals)
VARA	R	Hold Area	Output, variance of A array
VARB	R	Hold Area	Output, variance of B array

Common Blocks and Variables Referenced

None.

10.5 Subroutine PRINT

Program Narrative

Purpose

Prints contents of matrices (which are to be analyzed by other routines) along with basic comparison calculations such as averages, differences and proportions.

Assumptions

None.

Approach

For networks A and B, this routine prints the contents of the analysis matrices which are to be compared in other routines. It is printed in tabular form where each MOE, subinterval and link are headings. Averages, differences and proportions of the two networks are included within the tables.

Synopsis

The routine loops through all MOEs, links and subintervals and stores link-specific statistics of two networks. Basic comparison calculations, such as averages, differences and proportions are then computed for each of the networks. The results are printed in tabular form separating each MOE, link and subinterval for easy comparisons. The routine then loops through all MOEs and subintervals comparing the network wide statistics of the two analysis networks. The results are then printed in similar tabular form.

Models and Algorithms

None.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(27)	R	General Purpose Hold Area	For Network A: 1) contains the j^{th} MOE corresponding to the i^{th} link in the k^{th} subinterval 2) contains the i^{th} MOE averaged over subintervals and links
B(27)	R	General Purpose Hold Area	For Network B: 1) contains the j^{th} MOE corresponding to the i^{th} link in the k^{th} subinterval 2) contains the i^{th} MOE averaged over subintervals and links
E(27)	R	Hold Area	Contains the difference of each MOE for networks A and B
I	I	General Purpose Do-loop Index	
IASATP (5)	I	Constant	Used as data instruction to print heading as output for tables
IBLA	I	Constant	Used as data instruction to print heading output for tables
IFRS	I	Switch	(1,2) for (freeway, all links)
II	I	Hold Area	Upstream node of i^{th} link

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IPEE(8)	I	Constant	Used as data instruction to print heading as output for tables
IX	I	Hold Area	Used as lower bound in loop for write statements
IY	I	Hold Area	Used as upper bound in loop for write statements
IZZ	I	Switch	(1,2) write headings for sub-interval (1-13, 14-26)
J	I	General Purpose Do-loop Index	
JJ	I	Hold Area	Downstream node of i^{th} link
JZZ	I	Switch	(1,2) subinterval from (1-13, 14-26)
KOUNT	I	Hold Area	Used as counter
L	I	Do-loop Index	From 1 to LFRS (number of links)
LFRS	I	Hold Area	Number of links (either free-way or non-freeway)
MOE	I	Do-loop Index	To loop over all MOEs
MOEL (5,8,2)	I	Hold Area	Array used as data instruction to print headings as output for tables
NINT	I	Constant	Number of subintervals

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
NLS	I	Switch	(0,1) if link-by-link analysis (should be, should not be) performed
NST	I	Hold Area	Number of freeway links
R(27)	R	Hold Area	Contains the quotient of each MOE for network A and B

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/SAMP2/ ISKIP, LF, LIN, LR, LS, N,
NLSA, NLSA2, NT

/SAMP3/ DMAT, DMDT, DNTA, DNTB,
DONETA, DONETB

10.6 Subroutine RANK

Program Narrative

Purpose

RANK scans all the values of a vector and puts each in rank order where the smallest value is ranked 1 and the largest value is ranked N.

Assumptions

None.

Approach

The original vector is searched for successively larger elements as a new vector is created in rank order. If ties occur, they are located and their rank values are computed. For example, if two values are tied for sixth rank, they are assigned a rank of 6.5 $[(6+7)/2]$.

Synopsis

RANK loops over each of the N data points in vector A and ranks it in accordance to its value. The smallest value is ranked 1 and largest value is ranked N. If data points are tied, their rank is computed by an average of the ranks of the two values. Otherwise, the rank is stored and a new data point is investigated. If a data point has already been ranked, the next data point is investigated. Processing goes on until all data points in the vector have been ranked.

Models and Algorithms

Each data point is ranked by comparing it, in turn, with all values in the vector.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
EQUAL	R	Hold Area	Counts number of data points equal to value being investigated
I	I	Do-loop Index	Data point index, from 1 to N
J	I	Do-loop Index	Data point index, from 1 to N
P	R	Hold Area	Rank of tied data points
SMALL	R	Hold Area	Counts number of data points smaller than value being investigated
X	R	Hold Area	Set equal to particular value in vector to be ranked

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(1)	R	Hold Area	Input, vector of N values
N	I	Hold Area	Input, number of values
R(1)	R	Hold Area	Output, vector of length N. Smallest value is ranked 1. Largest is ranked N.

Common Blocks and Variables Referenced

None.

10.7 Subroutine READCL

Program Narrative

Purpose

Reads statistical data from statistical data tape and primes arrays.

Assumptions

None.

Approach

The routine reads the data tape created by POSPRO, transforms the simulation network data to create the analysis network and performs the necessary calculations to prime the analysis matrices.

Synopsis

The routine reads the output from the statistical data tape created by POSPRO at the end of a given subinterval. It then transforms the link-specific data into an analysis network by combining and deleting specified link. The appropriate calculations are performed to derive the MOEs necessary to prime the link-specific and network-wide analysis matrices. The DRWS routine is called several times to read data from one of the two (networks A and B) statistical data tapes.

Models and Algorithms

None.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	General Purpose Switch Do-loop Index	(1,2,3) for (freeway, ramp, surface) General purpose do-loop index
IB (8,NTOTL)	I	Hold Area	Parameter in argument list for DRWS routine. Contains array of data that has been read
IZ (8,NTOTL)	I	Hold Area	Equivalent to IB
M	I	Switch	(1,2) for (just free links, all links)
MOE	I	Do-loop Index	Loops over appropriate MOEs

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
DMAT (110,8,2)	R	Hold Area	Output, array that contains calculated MOE for (link, MOE type, subinterval)
DNT (6,26,2)	R	Hold Area	Output, array that contains calculated MOE for (MOE type, subinterval, network type--all link or just freeway)
INET	I	Switch	Input, (1,2) for networks (A,B)

Common Blocks and Variables Referenced

/SAMP2/ LF, LR, LS, N, NFILEA
/CONTRL/ IFN, IFORM, ILEN, IOP

10.8 Subroutine STAT

Program Narrative

Purpose

Subroutine to perform all statistical tests for each appropriate MOE on all link combinations and subintervals as well as for network-wide MOEs. Also prints results in tabular form.

Assumptions

None.

Approach

The routine proceeds through the link-specific and network-wide analysis matrices priming the analysis vectors. For each pair of vectors, a choice of statistical tests to be performed is made based on sample size. The appropriate routines are called and the results of these tests are printed.

Synopsis

The routine performs the following three groups of tests for appropriate MOEs (certain MOEs apply to only: all links, freeway links, non-freeway links and the entire network).

- 1) For each link, performs the indicated statistical tests comparing the results of Networks A and B, treating the subinterval values as points in a statistical distribution and prints the results.
- 2) For each subinterval, performs the indicated statistical tests comparing the results of Networks A and B, treating the individual link values as points in a statistical distribution and prints the results.

- 3) For the network values, performs the indicated statistical tests comparing the results of Networks A and B, treating the subinterval values as points in a statistical distribution and prints the results.

The following routines, depending on sample size, may be called to accomplish statistical tests--CALC, ANOVA, TTEST, WILCOX and UTEST

Models and Algorithms

None.

Date, Programmer

November 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A (220)	R	Hold Area	Particular MOE array from Network A to be analyzed
ABAR	R	Hold Area	Mean of A array
ASTR(5)	R	Hold Area	Data statement used for table printout
B (110)	R	Hold Area	Particular MOE array from Network B to be analyzed
BBAR	R	Hold Area	Mean of B
D (220)	R	Hold Area	Holds t-test statistic, Wilcoxon-t-statistic or Z-statistic
F	R	Hold Ara	F-test statistic

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IASATP (5)	I	Hold Area	Data statement used for table printout
IPEE (8)	I	Hold Area	Data statement used for table printout
ISPDY (5)	I	Hold Area	Data statement used for table printout
MOEL (5,8,2)	I	Hold Area	Array that contains various data statements used for table printout
MOETN (4,6)	I	Hold Area	Data statement used for table printout
SSB	R	Hold Area	Sum of squares between the two arrays
TT	R	Hold Area	Sum of squares within the two arrays
TT	R	Hold Area	T-statistic
TW	R	Hold Area	Wilcoxon t- or Z-statistic
VARA	R	Hold Area	Variance of A
VARB	R	Hold Area	Variance of B
Z	R	Hold Area	U-test or Z-statistic

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/SAMP2/ ISKIP, LF, LIN, LR, LS, N,
NLSA, NLSA2

/SAMP3/ DMAT, DMDT, DNTA, DNTB

10.9 Subroutine TTEST

Program Narrative

Purpose

Perform the paired t-test on the mean of a given pair of arrays to test whether the two samples are from the same population.

Assumptions

- The arrays are two random samples of the same size which are independent;
- The variance of each array is equal;
- Depends on assumption of normality and requires that all pairs be observed under the same condition.

Approach

The routine calculates the t-statistic, compares it against the t-table and determines its significance.

Synopsis

The two random samples to be tested are provided as input by the argument list of the calling statement. The routine follows the standard arithmetic operations to calculate the t-statistic. It then computes the number of degrees of freedom which is used as the pointer to the t-table which is imbedded within the routine. The level of significance is determined and that information, as well as the t-statistic is stored in the argument list. If the degrees of freedom are greater than 25, the degrees of freedom and their significance are interpolated.

Models and Algorithms

Described in Crow, E., et al,
"Statistical Manual," Dover Publications, Inc., New York,
1960, Chapter 2.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
DBAR	R	Hold Area	Average of the differences of Arrays A and B
EPS	R	Constant	Equals .00001. N is compared against EPS for default value of N
I	I	Do-loop Index	Set from I to N representing each value in the arrays
IDF	I	Hold Area	Degrees of freedom. Used as a pointer in the t-table
SIG (25,3)	R	Hold Area	Imbedded table of critical values of t in the t-table
STD	R	Hold Area	Standard deviation of the differences of the two arrays
SUMD	R	Hold Area	The sum of the differences of the two arrays
SUMDD	R	Hold Area	Used as numerator in calcu- lating the variance
T	R	Hold Area	The t-statistic

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
VAR	R	Hold Area	The variance of the differences of the two arrays
XDF	R	Hold Area	Same as IDF but a real number
XN	R	Hold Area	Number of samples in Arrays A and B

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A (1)	R	Hold Area	Input, vector of length N containing data from the first sample
B (1)	R	Hold Area	Input, vector of length N containing data from the second sample
D (1)	R	Hold Area	Output, contains the t-statistic
ISIG	I	Switch	Output, (1,2,3) indicates level of significance
NINT	I	Hold Area	Input, number of observation (subintervals) in each vector A and B

Common Blocks and Variables Referenced

None.

10.10 Subroutine UTEST

Program Narrative

Purpose

UTEST tests whether two independent groups are from the same population.

Assumptions

- Both samples are random samples from their respective populations.
- In addition to independence within each sample, there is mutual independence between the two samples.
- Both samples consist of continuous random variables. (A moderate number of two is tolerable.)

Approach

When at least ordinal measurement has been achieved, the Mann-Whitney U-test is used to test whether two independent groups have been drawn from the same population.

Synopsis

Subroutine UTEST tests whether two independent groups are from the same population by means of the Mann-Whitney U-test, given an input vector, A, with the smaller group preceding the larger group. The scores for both groups are ranked together in ascending order by calling subroutine RANK. Tied observations are assigned the average of the tied ranks.

The sum of the ranks in the larger group, R_2 , is calculated. The U-statistic is then computed. The significance of U is measured by Z. Z is calculated differently depending if the size of the larger sample, N_2 , is greater than or equal to 20. If Z is less than 20, $Z=u$. Otherwise, the standard deviation, S, is computed and Z is calculated using S. That is, if the

the number of degrees of freedom exceeds the table capacity, the U-statistic is transformed to a Z-statistic and a normal approximation is made. Z (or U) is finally compared to imbedded tabular values to determine whether the null hypothesis H_0 should be rejected. The null hypothesis is that the two samples have the same distribution.

Models and Algorithms

Described in Siegel, S., "Non-parametric Statistics for the Behavioral Sciences," McGraw-Hill, New York, 1956, Chapter 6, Pages 116 to 126.

Date, Programmer

October 1975. Don Goldman.

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
FN	R	Hold Area	The sum of the number of cases in both groups (N_1+N_2)
FNX	R	Hold Area	$N_1 \cdot N_2$
FN2	R	Hold Area	Number of cases in larger group
I	I	Do-loop Index	R array index, set to rank larger group
IDF	I	Hold Area	Set to locate table value imbedded in subroutine (degrees of freedom)
N	I	Hold Area	Same as FN but integer
N2	I	Hold Area	Number of cases in smaller group plus one (N_1+1)

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
R2	R	Hold Area	Sum of ranks in the larger group
S	R	Hold Area	Standard deviation
TABLE (18,2)	R	Hold Area	Imbedded table of critical values of U in the Mann-Whitney Test and Z.
U	R	Hold Area	Statistic used to test homogeneity of the two groups
UP	R	Hold Area	The U-statistic if $UP < U$ ($U = N1 \cdot N2 - UP$)
Z	R	Hold Area	Measure of significance of U in terms of normal distribution

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(1)	R	Hold Area	Input, vector of cases consisting of two independent groups. Smaller group precedes larger group. Length is $N1 + N2$
ISIG	I	Switch	Output, (0,1,2) level of significance indicator
N1	I	Hold Area	Input, number of cases in smaller group
N2	I	Hold Area	Input, number of cases in larger group
R(1)	R	Hold Area	Output, U (or Z) statistic

Common Blocks and Variables Referenced
None.

10.11 Subroutine WILCOX

Program Narrative

Purpose

Non-parametric method for testing the null hypothesis that two samples come from identical populations. Performed by use of the Wilcoxon matched pairs signed ranks test on a given pair of arrays.

Assumptions

Let the absolute difference for all matched pairs between the two scores equal $|D_i|$. Then

- Each D_i is a continuous random variable;
- The distributions of each D_i are symmetrical;
- The D_i s are mutually independent;
- The D_i s have the same median;
- The measurements scale for the D_i s is at least interval.

Approach

The routine calculates the Wilcoxon-t-statistic, compares it against the Wilcoxon-t-table, and determines its significance.

Synopsis

The routine first calculates differences of the two arrays. Subroutine RANK is called to rank these differences. The routine follows the standard arithmetic operations to calculate the Wilcoxon-t-statistic. It then computes the number of degrees of freedom which is used as a pointer to the Wilcoxon-t-table imbedded within the subroutine. If the number of degrees of freedom exceeds the table capacity (N greater than 20), the t-statistic is transformed to a Z statistic and a normal approximation is made. If N is less than 6, the test is not applicable. The level of significance is determined and the information along with the T (or Z) statistic is stored in the argument list.

Models and Algorithms

Described in Siegel, S., "Non-parametric Statistics for the Behavioral Sciences," McGraw-Hill, New York, 1956, Chapter 6, Pages 75 to 83.

Date, Programmer

October 1975. Don Goldman

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
BIG	R	Hold Area	Positive difference of matched pairs
DIF	R	Hold Area	Signed difference of matched pairs
E(110)	R	Hold Area	Vector of ranked differences
FK	R	Hold Area	Number of matched pairs
I	I	Do-loop Index	Set from 1 to N for all matched pairs
IDF	I	Hold Area	Degrees of freedom. Used as a pointer to the Wilcoxon-t-table
K	I	Hold Area	Number of matched pairs minus 1 (N-1)
L(110)	I	Hold Area	Array that stores signs of differences of matched pairs
S	R	Hold Area	Standard deviation used to compute Z statistic

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
SIG (21,3)	R	Hold Area	Imbedded table of critical values of t in the Wilcoxon test and Z value
SUMM	R	Hold Area	Sum of ranks of minus differences
SUMP	R	Hold Area	Sum of ranks of positive differences
V	R	Hold Area	Mean used to compute Z statistic
Z	R		Z statistic

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
A(1)	R	Hold Area	Input, vector of length, N, containing data from the first sample
B(1)	R	Hold Area	Input, vector of length, N, containing data from the second sample
D(1)	R	Hold Area	Output, Wilcoxon-t (or Z) statistic
ISIG	I	Switch	Output, (0,1,2,3) for level of significance indicator
N	I	Hold Area	Input, number of observations in each vector, A and B

Common Blocks and Variables Referenced

None.

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
KTEST	I	Hold Area	Flag (0,1) if detector (never, previously) actuated
KTI	I	Hold Area	Time of actuation, milliseconds
L	I	Hold Area	Detector type
LDET	I	Hold Area	Detector length
LDT	I	Hold Area	Elapsed time, milliseconds
LEND	I	Hold Area	Detector loop length
LSTN	I	Hold Area	Time of last actuation
ND	I	Hold Area	Detector number
NIZ	I	Hold Area	Index to previous occupancy data
NTEM	I	Hold Area	File number of detector data files
NUDE	I	Hold Area	Detector number
NUMBER	I	Hold Area	Number of actuations in a block of data
NUMDET	I	Hold Area	Number of detectors at a sta- tion
OCC	R	Hold Area	Percent occupancy at a detector
OCC2	R	Hold Area	Percent occupancy at a detector
VAL	R	Hold Area	Occupancy updating interval, milliseconds

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A4/DTCTR,IDET

/CONTRL/IFN,IFORM,IIDMOD,IIDOP,ILEN,IOP

/CRASH/IFIN,INTVAL,IOCC,IOCC2,IZERO,
KALL,KON,KPOINT,LASTOF,LASTON,LENTHV,
LOCATE,OCCS

/MTOC/TOCC

/PDATA/IRAW,ISAMP,LSTVU

REFERENCES

1. Payne, H.J., "Analysis and Evaluation of the California Logic for Incident Detection," USCEE Report 464, Final Report Part III, December 1973.
2. Payne, H. J. and Knobel, H. C., "Development and Testing of Incident Detection Algorithms," Volume 3, FHWA Report No. FHWA-RD-76-21, February 1976.
3. Payne, H. J., Goodwin, D. N. and Teener, M. D., "Evaluation of Existing Incident Detection Algorithms," FHWA Report No. FHWA-RD-75-39, February 1975.
4. Gazis, D. C. and Knapp, C. H., "On-line Estimation of Traffic Densities from Time-Series of Flow and Speed Data," Transportation Science, Volume 5, Number 3, August 1971.
5. Gazis, D. C. and Szeto, M. W., "Design of Density--Measuring Systems for Roadways," Transportation Research Record 495, 1974.
6. Nahi, N. E. and Trivedi, A. N., "Recursive Estimation of Traffic Variables: Section Density and Average Speed," Transportation Science, Volume 7, Number 3, August 1973.

ADDENDUM

DEVELOPMENT AND TESTING OF INTRAS,
A MICROSCOPIC FREEWAY SIMULATION MODEL

Volume IV: Program Documentation

Subroutine FXCHNL

Program Narrative

Purpose

FXCHNL locates a vehicle downstream of an input vehicle in its lane.

Assumptions

None.

Approach

The given vehicle's link, lane, and position are identified. The vehicle array is then scanned until a leader is found.

Synopsis

The input vehicle is located on a link and a lane. The vehicle array is scanned for a vehicle downstream of the given one in its link and lane. If one is found, its number is returned. If no vehicle is identified, a search is made for the link and lane immediately downstream of the given one. The last vehicle on that lane is returned. If no vehicle on that lane is found, a zero is returned as the lead vehicle number.

Models and Algorithms

None.

Date; Programmer

February 1978; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Loop over vehicle
ID	I	Hold area	Vehicle number
INDEX	I	Hold area	Index to VF array
IPOS	I	Hold area	Vehicle position
L	I	Hold area	Subject vehicle's link number
LA	I	Hold area	Subject vehicle's lane number
LD	I	Hold area	Downstream link number
LDTP	I	Hold area	Downstream link type
LND	I	Hold area	Downstream lane number

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IID	I	Hold area	Output, had vehicle number. Initialized to zero.
IIV	I	Hold area	Input, vehicle number whose leader is to be identified.

Common Blocks and Variables Referenced

/AI/ NLV
/SIZE/ MXVFR, NLFO

Subroutine FXCHNF

Program Narrative

Purpose

FXCHNF reestablishes a chain of vehicles for a given link and lane.

Assumption

None.

Approach

The routine locates vehicles on a given link and lane. The stack is sorted.

Synopsis

The vehicle array is searched for vehicles given on a link and lane. Once located, they are placed in a stack. Once the stack is filled with the number of vehicles required it is sorted by vehicle position. After this is done, the new chain is loaded into the INTRAS Database.

Models and Algorithms

None.

Date; Programmer

February 1978; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Hold area	Loop index over vehicles
ICAR	I	Hold area	Pointer to position in stock array
ID	I	Hold area	Lead vehicle number
IND	I	Hold area	Pointer to data item in COMMON
INew	I	Hold area	Stack pointer
ISTACK	I	Hold area	Stack holding vehicle numbers
KCAR	I	Hold area	Car pointer to VF array

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ILINK	I	Hold area	Link number
ILANE	I	Hold area	Lane number

Common Blocks and Variables Referenced

/A1/ NLV
/A2/ ENDNDS
/CONTRL/ MXINIT,
/SIZE/ MXVFR, NLFO, NVFO,
/S2/ ITIME

Subroutine DMPOUT

Program Narrative

Purpose

DMPOUT dumps the contents of common storage to the periodic or one-time restart files.

Assumptions

None.

Approach

All data is written using unformatted writes.

Synopsis

At a given time either a periodic dump may be made, a one-time dump may be made, or both may be made. Once the output file has been established, the Common Storage area is dumped. Routines are called to dump the program peripheral files if they exist.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
K	I	Do-loop index	From 1 to 2 for up to two types of restart dumps
KFN	I		Output file name KFN = {IFTSTR} if dump is {one time only} {periodic}

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/	MNLV, NLV
/A2/	NTOTL, IVFEET, ENDNDS
/A3/	NTOTN, SIGI
/A4/	NDET, IDET, DTCTR
/A5/	NINCI, INCID
/A6/	MNODE, IANOD, MANOD, SIG, NPHS, SIGT, IACPAR
/A7/	NS, SNODE
/A8/	JMPG, ILT, IRT, IALAG, JERK, SPLPCT, CLFN, LSLAG, ITGAP, IAMBR, IGAP, MINAC, VFH, VFHF, FOLK, LSTTIM, LIMSPD, MAXAC, CSTDCL, LNMHSP, ITPCT, TLNK, IPACK
/A9/	CHEK78
/A10/	VPROC, MNJVR, LTRNV, THRUV, RTRNV, VCONT
/A11/	ISTATN, WSTATN, LSTATN
/A12/	IFCOEF, IGRVAL
/ACT0/	MACR, NNCR, ISTDET
/ACT1/	NAFZ
/ACT2/	NAAFZ
/ACT3/	NATMR
/ACT4/	NADET

/ACT7/ NALNK
 /ACT8/ NACDS
 /ACT9/ NACYC
 /ACT10/ NACT
 /ACT12/ NASTAT
 /ACT20/ SIGA
 /CFVEL/ JFVEL, MFJEL, NJ23, IPTAB, NFLG
 /CONTRL/ I99LAS, IRSCT, WEXCAL, LASJBF,
 IFN, IOP, IFORM, ILEN, ISCT, INLOC,
 IPL, ICTYPE, ICSIN, ICSRN, IINOT,
 IPTOT, ISTOR, ISTAT, IIWIN, IINCON,
 IPT, ICARD, ICDCT, IBUF, MINSTS,
 MAXSTS, RNSEED, IDRUN, MXINIT,
 IFILGO, IGO, ISTFLG, NTRN, ITPOP,
 ITPLNK, ICPOP, ICPLNK, ICPSP,
 IIDOP, IIDLNK, NOT, IDMN, NPRINT,
 ISECI, IDSEC, IODFLG, ILSUB, IIDMOD
 /CRASH/ KON, LASTON, LASTOF, IOCC, IOCC2,
 OCCS, OCCOLD, MAXCEL, INTVAL,
 IJPDAT, KACC, IZCRO, IFIN, LINTHV,
 IPPT, IMMOE, INCALG, MOEALG,
 EPARAM, MPARAM, NSECTL, MAXSTA,
 LNKSTA, NUMSTN, FPOINT
 /DBLEXP/ SMTH1, SMTH2, YERR, ABSD, SSQ
 /ERTRAN/ IER, NPAR, IEPAR, IERCT
 /FORNC/ ICALL, NUMBER, IINC, LNJDE, LLEND,
 LLEDET, LLNLEN, VELOLD, ACCOLD,
 ISTART, ISTEP, LENYH, IPOS1, ISPOI,
 LINKI, LANEI, IPOS2, ISPO2, LINK2,
 LANE2, NCHNG, ICVR, LINEON, INCTIM,
 IVNUM
 /FPAR/ INCH, NCH, INRAMP, IRF, IORD,
 MINH DY, IP, MNAC, ISENS, LAGA,
 LAGB, KET, LAG
 /FREDAT/ FRHDWY, FRSPED, FRQONT, FRFLAG,
 FRTIM
 /IGEAD/ IHEAD
 /INCDMP/ NJMOJT, LCIN, IIDVMY
 /IPLTN/ IPLTN
 /LASDCH/ LADISH
 /NFILTS/ NDSCHG, NDSCHO, NENTRO, NLASEQ
 /NIDTSV/ NIDTSV
 /NRGY/ JVACCP, JVACC
 /OWVEH/ NORG, IORG, IRV, IREN

/P1/	MAMES
/RANK/	NFENT, IFRANK
/S1/	MESERR
/S2/	IDTF, IDTS, ITIME, ITIMEF, IDT, IFDT, NIDT, IPSL, IEXCS, MCYCL, IMCYCL, IREVP, ICOINT, ICYCPO, INTSTO, ITRINT, IHOURS, IMINS, ISECNS
/S3/	ISSEQ, ISRET
/SIZE/	NLFR, NLRA, NLSJ, NVFR, NVRA, NVSU, NLFO, NLRO, NLDO, NVFO, NVRO, NVSO, MXLFR, MXLRA, MXLSU, MXVFR, MXVRA, MXVSU, IS, IWDPAK, IWDINO, NWDS
/VBUF/	NF, NRL, NRH, NFL, NFA
/VPOINT/	IFPT, IRPT, ISPT
/MISC/	ICASNO
/RSTR/	IFTSTRW, IFDLT, ITPD

Subroutine DMPIN

Program Narrative

Purpose

DMPIN reloads the COMMON storage area from the restart file.

Assumptions

A restart file is assumed to be resident on file IFSTRT.

Approach

All data is read using unformatted reads.

Synopsis

All common storage areas are restored from the restart file. After this is complete, the input data files, detector data files and the fuel data files are restored and positioned properly. Finally, a control options table is printed.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IHO	I	Hold area	Current time, hours
IMO	I	Hold area	Current time, minutes
ISO	I	Hold area	Current time, seconds
ISR	I	Hold area	Temporary storage
KFN	I	Hold area	File where restart date resides

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/A1/	MNLV, NLV
/A2/	NTOTL, IVFEET, ENDNDS
/A3/	NTOTN, SIGI
/A4/	NDET, IDET, DTCTR
/A5/	NINCI, INCID
/A6/	MNODE, IANOD, MANOD, SIG, NPFS, SIGT, IACPAR
/A7/	NS, SNODE
/A8/	JMPG, ILT, IRT, IALAG, JERK, SPLPCT, CLFN, LSLAG, ITGAP, IAMBR, IGAP, MINAC, VFH, VFHF, FOLK, LSTTIM, LIMSPD, MAXAC, CSTDCL, LNMNSP, ITPCT, TLNK, IPACK
/A9/	CHEK78
/A10/	VPROC, MNJVR, LTRNV, THRUV, RTRNV, VCONT
/A11/	ISTATN, WSTATN, LSTATN
/A12/	IFCOEF, IGRVAL
/ACT0/	MACR, NNCR, ISTDET
/ACT1/	NAFZ
/ACT2/	NAAFZ
/ACT3/	NATMR
/ACT4/	NADET
/ACT7/	NALNK
/ACT8/	NACDS

/ACT9/	NACYC
/ACT10/	NACT
/ACT12/	NASTAT
/ACT20/	SIGA
/CFUEL/	UFUEL, MFUEL, NU23, IPTAB, NFLG
/CONTRL/	I99LAS, IRSCT, WEXCAL, LASUBF, IFN, IOP, IFORM, ILEN, ISCOT, INLOC, IPL, ICTYPE, ICSIN, ICSRN, IINOT, IPTOT, ISTOR, ISTAT, IIWIN, IINCON, IPT, ICARD, ICDCT, IBUF, MINSTS, MAXSTS, RNSEED, IDRUN, MXINIT, IFILGO, IGO, ISTFLG, NTRN, ITPOP, ITPLNK, ICPOP, ICPLNK, ICPSP, IIDOP, IIDLNK, NOT, IDMN, NPRNT, ISECI, IDSEC, IODFLG, ILSUB, IIDMOD
/CRASH/	KON, CASTON, LASTOF, IOCC, IOCC2, INPOS, INCODE, LOCATE, OCCS, OCCOLD, MAXCEL, INTVAL, IUPDAT, KALL, IZCRO, IFIN, LENTHV, IPPT, IMMOE, INCALG, MOEALG, EPARAM, MPARAM, NSECTL, MAXSTA, LNKSTA, NUMSTN, KPOINT
/DBLEXP/	SMTH1 SMTH2, YERR, ABSD, SSQ
/ERTRAN/	IER, NPAR, IEPAR, IERCT
/FORNC/	ICALL, NUMBER, IINC, LNUDE, LLEND, LLDET, LLNLEN, VELOLD, ACCOLD, ISTART, ISTEP, LENYH, IPOS1, ISPO1, LINK1, LANE1, IPOS2, ISPO2, LINK2, LANE2, NCHNG, ICVR, LINEON, INCTIM, IVNUM
/FPAR/	INCH, NCH, INRAMP, IRF, IORD, MINHDY, IP, MNAC, ISENS, LAGA, LAGB, KET, LAG
/FREDAT/	FRHDWY, FRSPED, FRQONT, FRFLAG, FRTIM
/IHEAD/	IHEAD
/INCDMP/	NVMOUT, LCIN, IIDUMY
/IPLTN/	IPLTN
/LASDCH/	LADISH
/NFILTS/	NDSCHG, NDSCHO, NENTRO, NLASEQ
/NIDTSV/	NIDTSV
/NRGY/	JVACCP, JVACC
/OWVEH/	NORG, IORG, IRV, IREN
/P1/	NAMES
/RANK/	NFENT, IFRANK

/S1/	MESERR
/S2/	IDRF, IDTS, ITIME, ITIMEF, IDT, IFDT, NIDT, IPSL, IEXCS, MCYCL, IMCYCL, IREVP, ICOINT, ICYCPO, INTSTO, ITRINT, IHOURS, IMINS, ISECNS
/S3/	ISSEQ, ISRET
/SIZE/	NLFR, NLRA, NLSV, NVFR, NVRA, NVSV, SLFO, NLRO, NLSO, NVFO, NVRO, NVSO, MXLFR, MXLRA, MXLSV, MXVFR, MXVRA, MXVSU, IS, IWDPAK, IWDINO, NWDS
/VBUF/	NF, NRL, NRA, NFL, NFA
/VPOINT/	IFPT, IRPT, ISPT
/MISC/	ICASNO
/RSTRT/	IFSTRT

Subroutine LOCTST

Program Narrative

Purpose

LOCTST determines if there are any required calls to the low activity overlay after the current time step.

Assumptions

None.

Approach

The current simulated time is compared to the time for the next call to LOCON for a given purpose. If a call is required, it is set up.

Synopsis

The following reasons to call LOCON are tested sequentially:

1. Time to test the end of initialization.
2. Time for occupancy or incident updates
3. Time for contour plots
4. Time for standard output
5. Time for intermediate output
6. Time for restart dumps.

If it is time to call LOCON for any of these reasons, the type of call is stored and a flag is set to return control to the SIFT overlay to call LOCON.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Used to zero array
ITEMP	I	Hold area	Temporary storage
ITT	I	Hold area	Temporary storage
J	I	Hold area	Temporary storage
JNTV	I	Hold area	Time interval for occupancy updates

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ ICPOP, IDMN, IDSEC, IFILGO,
ISECI, MXINIT, NPRNT
/CRASH/ INTVAL
/FORINC/ LINEON
/RSTRT/ IRDMP, ITDELT, ITPD, ITSTAR
/SEQU/ IFSEQ, JFSEQ
/S2/ ICOINT, ICYCPO, IDT, IDTS, IMCYCL,
ITIME, ITIMEF, MCYCL, NIDT
/S3/ ISRET, ISSEQ

Subroutine POSIT

Program Narrative

Posit positions the input data file.

Assumptions

None.

Approach

The file is read until after data for the current subinterval is found.

Synopsis

The file is rewound and read until the type 60 card is found. A subinterval counter is updated. If the count is the same as the current subinterval the routine returns control to the calling program.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IB	I	Hold area	Dummy argument
ICOUNT	I	Hold area	Count of subintervals

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, IFORM

Subroutine DOIDFL

Program Narrative

Purpose

DOIDFL dumps the input card file stored on File 13.

Assumptions

File 13 has the input card file.

Approach

The input file is copied to the appropriate restart files and then repositioned.

Synopsis

The input file is rewound and copied card image by card image to the appropriate restart files. When the end of file is encountered, endfiles are written on the restart files and the input file is repositioned to continue simulation.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

None.

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IBUF, IFN, IFORM, OLEN, IOP
/RSTRT/ IFDLT, IFTSTR, ITPD

Subroutine DIIDFL

Program Narrative

Purpose

DIIDFL restores the input data file from the restart tape.

Assumptions

The restart tape is available.

Approach

The restart tape is read and data is transferred to the input data file.

Synopsis

The restart tape is read one card image at a time. The image is printed out and written in the input data file. This continues until an endfile is read. The input data file is then positioned.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Used in read
IBUF	I	Hold area	Card image buffer
IFN	I	Hold area	Input data file number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/RSTRT/ IFSTRT

Subroutine DODDF

Program Narrative

Purpose

DODDF dumps the online and offline detector data files to restart tape.

Assumptions

None.

Approach

Data is transferred from the detector files to the restart tape.

Synopsis

The procedure is identified for online and offline detector tapes. An end of file is placed on the detector tape and it is rewound. Data is then transferred record by record to either the periodic restart tape, the one-time restart tape, or both of them as required. When an end of file is encountered on the detector tape, it is repositioned for further simulation by a backspace.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
IBOUT	I	Hold area	Detector data buffer
NTOFF	I	Hold area	"Offline" detector file number
NTON	I	Hold area	"Online" detector file number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/ IFN, IFORM, IINOT, ILEN, IOP
/FORINC/ LINEON
/RSTRT/ IFDLT, IFTSTR, ITPD

Subroutine DDDF

Program Narrative

Purpose

DDDF restores all detector files from the restart tape.

Assumptions

The restart tape is available.

Approach

The restart tape is read and data is transferred to the appropriate detector files.

Synopsis

The restart tape is read one record at a time. The data is transferred to the appropriate detector files. This continues until an end of file is encountered. The procedure is repeated for offline detector files as well as online files.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Implied loops
IBOUT	I	Hold area	Detector data buffer
NTOFF	I	Hold area	Offline detector file number
NTON	I	Hold area	Online detector file number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CONTRL/	IFN, IFORM, IINOT, ILEN, IOP
/FORINC/	LINEON
/RSTRT/	IFSTRT

Subroutine DOFDF

Program Narrative

Purpose

DOFDF dumps the fuel data files to the restart tape.

Assumptions

None.

Approach

Data is transferred from the fuel files to the restart tape.

Synopsis

An end of file is written to the fuel files and they are rewound. They are then copied to the restart tape record by record until the end of files are encountered. At that time the fuel files are repositioned to continue simulation.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Set to 22 and/or 23
J	I	Hold area	Fuel data file number
JVACC	I	Hold area	Fuel data buffer
JVACCP	I	Hold area	Number of entries in fuel buffer
K	I	Hold area	Fuel data file number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CFUEL/ JFUEL, NU23
/CONTRL/ IFN, IFORM, ILEN, IOP
/RSTRT/ IFDLT, IFTSTR, ITPD

Subroutine DIFDF

Program Narrative

Purpose

DIFDF restores the fuel data files from the restart tape.

Assumptions

The restart tape is available.

Approach

The restart tape is read and data is transferred to the appropriate fuel files.

Synopsis

The restart tape is read two records at a time and the data is used to load the appropriate fuel files. Upon encountering an end of file on the restart tape, all fuel files are properly positioned.

Models and Algorithms

None.

Date; Programmer

November 1977; Reuben Goldblatt

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
I	I	Do-loop index	Loop over fuel file numbers
J	I	Hold area	Fuel file number
JVACC	I	Hold area	Fuel data buffer
JVACCP	I	Hold area	Number of entries in fuel buffer
K	I	Hold area	Fuel file number

Arguments Passed to Subroutine

None.

Common Blocks and Variables Referenced

/CFUEL/ JFUEL, NU23
/CONTRL/ IFN, IFORM, ILIN, IOP
/RSTRT/ IFSTRT

Subroutine STOPFL

Program Narrative

Purpose

STOPFL sets up a virtual lane blockage on a freeway link having an exit ramp as a destination. The lane blockage is in the through lane next to an auxillary lane which leads to the exit ramp and is perceived only by the exiting vehicle furthest downstream in the lane. The purpose of this virtual incident is to simulate the behavior of an exiting vehicle slowing down to force its way into the auxillary lane leading to the exit.

Assumptions

It is assumed that the vehicle parameter vector has been stored in the NF array, that the link parameter vector has been stored in the IPACK array, and that the link has an exit ramp destination.

Approach

STOPFL will set up a virtual incident to slow down the leading exiting vehicle if BLKLN has not been set. If BLKLN has been set, the routine will increment the incident location by 5 feet

Synopsis

First STOPFL checks to see if BLKLN has already been set. If so, a transfer is made to increment the position of the virtual incident if the flagged vehicles' speed is less than 7 fps. If BLKLN is zero, the routine checks whether the subject vehicle is already in an exiting lane. If so, a RETURN is made to FMOVE. If not, the routine checks whether this is a left or right exit. In either case, a check is made to determine whether the vehicle is in a lane next to an exiting lane. If not, a RETURN is made to FMOVE. If so, a virtual lane blockage for the subject vehicle only in the lane is set up. The position of the virtual blockage is: (1) the distance downstream from the vehicles' current position which is required for the vehicle to decelerate to a stop at a rate of -10 fps if the auxillary lane is a full lane; (2) the distance downstream from the vehicles' current position to the tail of the auxillary lane plus 100 feet if the auxillary lane is a deceleration lane.

Models and Algorithms

None

Date, Programmer

June 1980 S. Cohen

Program Variables

Local Variables

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition, Initialization and Modification</u>
ILANE	I	Hold area	Lane occupied by vehicle M
ILINK	I	Hold area	Link occupied by vehicle M
ILL	I	Hold area	Exiting lane number
KTYP	I	Switch	Type of link lane ILL discharges into; = (1,≠1) if (freeway, not freeway)
LNL	I	Hold area	Length of link ILINK
LNLG	I	Hold area	Length of lane ILL
LTHRU	I	Hold area	Lane fed by lane ILL
M	I	Hold area	Vehicle number of subject vehicle
ND1	I	Hold area	Distance from which M to tail of auxillary lane ILL + 100 feet
ND2	I	Hold area	Distance to decelerate vehicle M to stop at -10fps
NDST	I	Hold area	Distance from upstream end of link to virtual incident
NEXL	I	Hold area	Downstream link fed by ILINK
NKTYP	I	Switch	Link type of link ILINK. Should be 1 for freeway link

Arguments Passed to Subroutine

<u>Name</u>	<u>Type</u>	<u>Description</u>	<u>Definition Initialization and Modification</u>
IIJ	I	Counter	Counter for number of incidents on link IILINK in lane ILANE
IILANE	I	Hold area	Lane occupied by vehicle MM
IILINK	I	Hold area	Link occupied by vehicle MM
KFLM	I	Switch	Code 1 if virtual incident has been set. Remains zero if virtual incident not set.
MM	I	Hold area	Vehicle number of subject vehicle
MNKTYP	I	Switch	Code: (1, ≠ 1) if IILINK (freeway, not freeway) link

Common Blocks and Variables Referenced

/A8/IPACK

/BLKLN/NVSTP

/VEER/LNEND

/VBUF/NF