PATROL CAR ALLOCATION MODEL: PROGRAM DESCRIPTION

PREPARED FOR THE OFFICE OF POLICY DEVELOPMENT AND RESEARCH, DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT,

AND FOR THE NATIONAL INSTITUTE OF LAW ENFORCEMENT AND CRIMINAL JUSTICE, LEAA, DEPARTMENT OF JUSTICE

JAN M. CHAIKEN
PETER DORMONT

R-1786/3-HUD/DOJ
SEPTEMBER 1975

THE NEW YORK CITY RAND INSTITUTE
PATROL CAR ALLOCATION MODEL: PROGRAM DESCRIPTION

PREPARED FOR THE OFFICE OF POLICY DEVELOPMENT AND RESEARCH, DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT,

AND FOR THE NATIONAL INSTITUTE OF LAW ENFORCEMENT AND CRIMINAL JUSTICE, LEAA, DEPARTMENT OF JUSTICE

JAN M. CHAIKEN
PETER DORMONT

R-1786/3-HUD/DOJ
SEPTEMBER 1975

THE NEW YORK CITY RAND INSTITUTE
This report provides installation instructions and an annotated program listing for a computer program written in the FORTRAN language, designed to assist police departments in determining the number of patrol cars to have on duty in each geographical command at different times of the day and week. The program—called the Patrol Car Allocation Model—is described in two companion reports:

- R-1786/1, Patrol Car Allocation Model: Executive Summary

The first of these, written for police department administrators and planning officials who wish to understand how the Patrol Car Allocation Model can be used in policy analysis, serves as the Summary of both the present volume and the User's Manual. The second provides all the information needed to use the program once it has been installed on a computer system.

The current report (R-1786/3) is written for data processing personnel. Most users will want to read only the first two chapters, which describe installation procedures, file formats, core requirements, and minor program modifications. For the benefit of users who may wish to make substantial modifications, the remainder of the report contains complete documentation, including a program listing with a detailed description of each subprogram.* A separate program used to calculate some of the data input for the Patrol Car Allocation Model is also described and listed in Appendix B, coauthored by David Jaquette.

Preparation of this report was supported by the Office of Policy Development and Research of the U.S. Department of Housing and Urban Development (HUD) under contract H-2164 with The New York City-Rand Institute. Among the objectives of this HUD contract are the development, field testing, and documentation of methods to improve resource

*The listings are printed 95 percent of actual size, in order to meet printing space constraints.
allocation procedures in municipal emergency service agencies in the United States.

Design of the Patrol Car Allocation Model was partially funded by HUD and partially by the National Institute of Law Enforcement and Criminal Justice under grant 75NI-99-0012 to The Rand Corporation.

This report is one of a series that documents several different deployment models for police, fire, and ambulance agencies. Further information can be obtained by writing to the addresses in Appendix D.
CONTENTS

PREFACE ................................................................. iii
INDEX OF PROGRAM SEGMENTS ...................................... vii
TABLES ........................................................................ ix
GLOSSARY ....................................................................... xi

Chapter
I. PROGRAM INSTALLATION ............................................. 1
   Introduction ................................................................ 1
   PCAM Source Language and Compilation ..................... 2
   File Structure and Conventions ................................ 3
   Storage Allocation .................................................. 5
   Core Requirements .................................................. 7
   Minor Program Modifications ................................... 8
   Costs ...................................................................... 9

II. PCAM DATA FILE FORMAT .......................................... 11

III. ALGORITHMS FOR CONVERSION OF ALLOCATIONS BETWEEN TOURS AND BLOCKS .............................................. 16
   Conversion of Tour Allocations to Block Allocations ...... 16
   Conversion of Block Allocations to Tour Allocations ...... 16

IV. INTERNAL DATA STRUCTURES ..................................... 19
   Storage Management ............................................... 19
   Table Pointers ........................................................ 20
   Data Storage .......................................................... 24

V. LISTING AND DESCRIPTION OF THE PCAM FORTRAN PROGRAM ......................................................... 28
   MAIN Program ......................................................... 28
   Subroutine INIT ....................................................... 31
   Subroutine GETBOT ................................................ 35
   Subroutine SCAN ................................................... 36
   Subroutine GETTKN ................................................ 41
   Function LKF1 ........................................................ 45
   Function LKP8 ........................................................ 46
   Subroutine MOVE .................................................... 47
   Subroutine GETTOP ................................................ 48
   Subroutine READ ..................................................... 49
   Subroutine GTDSC ................................................... 57
   Subroutine MRCORD ............................................... 60
   Subroutine DERIVE ................................................ 61
   Subroutine SBLACT ................................................ 65
   Subroutine SBLEF ................................................... 67
   Subroutine LIST ...................................................... 68
   Subroutine SETWFL ................................................. 72
Function NXPCT ........................................ 75
Function NXDAY ........................................ 77
Function NXTOUR ....................................... 78
Subroutine DISP ......................................... 80
Subroutine DSPPDT ....................................... 83
Subroutine DSPDTP ....................................... 86
Subroutine ZERO .......................................... 90
Subroutine TITLE ......................................... 91
Subroutine COMPTB ....................................... 92
Subroutine PRTBL ......................................... 97
Subroutine TOTAL ........................................ 98
Function AVTT ........................................... 100
Function OBJF1 .......................................... 102
Function QUEUE .......................................... 103
Function OBJF2 .......................................... 104
Function OBJF3 .......................................... 106
Subroutine SET ......................................... 108
Subroutine MEET ......................................... 112
Subroutine STRCAR ...................................... 117
Function KNSTR .......................................... 121
Subroutine CKOV R ....................................... 124
Subroutine ADDALC ...................................... 125
Subroutine SBLOBJ ....................................... 131
Subroutine STDF .......................................... 132
Subroutine ADJUST ...................................... 134
Subroutine STROBJ ...................................... 137
Function OBJFUn ......................................... 139
Subroutine ADDCAR ...................................... 141
Subroutine WRITE ........................................ 145
Subroutine SKIP .......................................... 150
Function CEIL ........................................... 151
BLOCK DATA ............................................... 152

Appendix
A. DEMONSTRATION DATA BASE .......................... 155
B. PROGRAM FOR ESTIMATING THE RELATIONSHIP BETWEEN CFS UNAVAILABILITIES AND NON-CFS UNAVAILABILITIES .......... 161
C. PROGRAM CROSS-REFERENCE TABLE ................ 170
D. ADDRESSES FOR FURTHER INFORMATION .............. 172
# INDEX OF PROGRAM SEGMENTS

<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDALC</td>
<td>125</td>
</tr>
<tr>
<td>ADDCAR</td>
<td>141</td>
</tr>
<tr>
<td>ADJUST</td>
<td>134</td>
</tr>
<tr>
<td>AVTT</td>
<td>100</td>
</tr>
<tr>
<td>BLOCK DATA</td>
<td>152</td>
</tr>
<tr>
<td>CEIL</td>
<td>151</td>
</tr>
<tr>
<td>CKOVIR</td>
<td>124</td>
</tr>
<tr>
<td>COMPTB</td>
<td>92</td>
</tr>
<tr>
<td>DERIVE</td>
<td>61</td>
</tr>
<tr>
<td>DISP</td>
<td>80</td>
</tr>
<tr>
<td>DSPDTP</td>
<td>86</td>
</tr>
<tr>
<td>DSPFDT</td>
<td>83</td>
</tr>
<tr>
<td>GETBOT</td>
<td>35</td>
</tr>
<tr>
<td>GETTKN</td>
<td>41</td>
</tr>
<tr>
<td>GETTOP</td>
<td>48</td>
</tr>
<tr>
<td>GTDSPC</td>
<td>57</td>
</tr>
<tr>
<td>INIT</td>
<td>31</td>
</tr>
<tr>
<td>KNSTR</td>
<td>121</td>
</tr>
<tr>
<td>LIST</td>
<td>68</td>
</tr>
<tr>
<td>LKP1</td>
<td>45</td>
</tr>
<tr>
<td>LKP8</td>
<td>46</td>
</tr>
<tr>
<td>MAIN</td>
<td>28</td>
</tr>
<tr>
<td>MEET</td>
<td>112</td>
</tr>
<tr>
<td>MOVE</td>
<td>47</td>
</tr>
<tr>
<td>MRGORD</td>
<td>60</td>
</tr>
<tr>
<td>NXDAY</td>
<td>77</td>
</tr>
<tr>
<td>NXPCT</td>
<td>75</td>
</tr>
<tr>
<td>NXTOUR</td>
<td>78</td>
</tr>
<tr>
<td>OBJF1</td>
<td>102</td>
</tr>
<tr>
<td>OBJF2</td>
<td>104</td>
</tr>
<tr>
<td>OBJF3</td>
<td>106</td>
</tr>
<tr>
<td>OBJFUN</td>
<td>139</td>
</tr>
<tr>
<td>QUEUE</td>
<td>103</td>
</tr>
<tr>
<td>PRTBL</td>
<td>97</td>
</tr>
<tr>
<td>READ</td>
<td>49</td>
</tr>
<tr>
<td>SBLACT</td>
<td>65</td>
</tr>
<tr>
<td>SBLEF</td>
<td>67</td>
</tr>
<tr>
<td>SLOBJ</td>
<td>131</td>
</tr>
<tr>
<td>SCAN</td>
<td>36</td>
</tr>
<tr>
<td>SET</td>
<td>108</td>
</tr>
<tr>
<td>SETWFL</td>
<td>72</td>
</tr>
<tr>
<td>SKIP</td>
<td>150</td>
</tr>
<tr>
<td>STRCAR</td>
<td>117</td>
</tr>
<tr>
<td>STRDF</td>
<td>132</td>
</tr>
<tr>
<td>STROBJ</td>
<td>137</td>
</tr>
<tr>
<td>TITLE</td>
<td>91</td>
</tr>
<tr>
<td>TOTAL</td>
<td>98</td>
</tr>
<tr>
<td>WRITE</td>
<td>145</td>
</tr>
<tr>
<td>ZERO</td>
<td>90</td>
</tr>
</tbody>
</table>
GLOSSARY

ALGORITHM
A procedure for performing a calculation.

ALLOCATE
1. Assign a certain number of cars to each shift.
2. Divide a fixed total number of car-hours among shifts.

AMPERAND (&)
At the end of a line of PCAM instructions, signifies that the com-
mmand continues on the following line.

ASTERISK (*)
1. At the start of a line of output from the DISP command, indicates
that the tour is overlaid by another tour.
2. In output tables, indicates a limiting constraint.
3. In input commands, represents the current number of car-hours
allocated.

AVAILABLE
1. Ready to be dispatched to a call for service.
2. Not engaged in cfs work or non-cfs work.
3. On preventive patrol.

BATCH
A mode of operating a computer program in which all instructions are
prepared on cards or other input device prior to program execution,
and output is received later from a high-speed printer. Contrasted
with INTERACTIVE.

BLOCK, TIME
A period of time (whole number of hours) over which the number of
patrol cars on duty does not change. One or two time blocks con-
istute a tour.

CALL RATE
Average number of calls for service received per hour.

CALL RATE PARAMETER
A parameter for each day in each precinct. When multiplied by the
hourly call-rate factor, gives the expected number of calls for
service in the hour.

CAR (see PATROL CAR)

CAR-HOUR (ACTUAL)
One patrol car on duty for one hour.
CAR-HOUR (EFFECTIVE)
   One hour spent by one patrol car on any activities other than non-cfs work.

CFS
   Call for service.

CFS WORK
   1. All activities of a patrol car from the time it is dispatched to
      a call for service until the time it is available again for
      dispatch.
   2. Number of car-hours spent on such activities.

CFS WORKLOAD
   1. Loosely speaking, the extent to which cfs work is a burden on a
      patrol car.
   2. Technically, the number of car-hours of cfs work in a given
      period of time.

COMMAND
   1. An instruction to the PCAM program.
   2. An administrative unit in a police department that is supervised
      by a superior officer. (Used in the expression geographical
      command.)

CONSTRAINING MEASURE
   Same as LIMITING CONSTRAINT.

CONSTRAINT
   A number specified as the largest or smallest value permitted for a
   performance measure.

CURRENT-DATA
   Some or all of the data in DATABASE, which have been read into the
   computer memory by a READ command and are used and/or modified by
   PCAM commands.

DATABASE
   The data prepared by the user for input into PCAM.

DAY
   A 24-hour period used for organizing PCAM data. Not necessarily a
   calendar day.

DELAY, TOTAL
   Sum of queuing delay and travel time. (Same as TOTAL RESPONSE TIME.)

DELIMITER
   Any character other than a letter, digit, parenthesis, asterisk,
   hyphen, period, or ampersand. Examples of delimiters are blanks,
   commas, colons, and equal signs.
DESCRIPTIVE MODE
Capability to calculate and display performance measures by time of
day and geographical command when the numbers of patrol cars on duty
in each shift have been specified.

DIVISION
A combination of precincts. Some police departments use the word
"division" for a precinct. This is permitted in PCAM by changing
the keyword PRECINCT.

EFFECTIVE CAR
The equivalent of a patrol car that does not engage in any non-cfs
work.

EXPONENTIALLY DISTRIBUTED
A random variable T is exponentially distributed if there is a param-
eter $\mu$ such that

$$\text{Prob}(T > t) = e^{-\mu t}.$$  

The mean of T is $1/\mu$. The assumption that service times for calls
to the police are exponentially distributed is not verified by data,
but the assumption is technically necessary in PCAM. (This is a
source of PCAM's simplicity.)

FIELDED
In the field. A patrol car is fielded if it is on duty.

FILLER WORD
One of the following words, which may be entered in a PCAM command
if desired, but will be ignored by the program: FOR, CAR, HOUR,
HOURS, TO, ON, BY, DATA.

HOURLY CALL RATE FACTOR
A parameter for a single hour in a single precinct. When multiplied
by the call rate parameter for the day, gives the expected number of
calls in the hour.

HOURLY SERVICE TIME FACTOR
A parameter for a single hour in a single precinct. When multiplied
by the service time parameter for the day, gives the expected service
time (in minutes) for calls received during the hour.

INTERACTIVE
A mode of operating a computer program whereby the user enters in-
structions at a terminal and receives output immediately at the same
terminal. Contrasted with BATCH.

KEYWORD
A character string that has a special meaning to the PCAM program.
These are either filler words or one of the following: DAY, P, C,
T, F, ADD, ALOC, DSP, END, LIST, MEET, READ, SET, WRITE, TOU (or
a substitute provided by the user), DIVISION (or a substitute),
PRECINCT (or a substitute).
LIMITING CONSTRAINTS
When meeting constraints, the particular performance measures whose
constrained values lead to a need for the largest number of patrol
cars. (If these constraints were eliminated, a smaller number of
patrol cars would meet all the constraints.)

LIST
Command that causes PCAM to print out the values of the data items
associated with all precincts, days, and tours within its scope.

MINIMUM ALLOCATION
The smallest whole number of actual patrol cars that can be assigned
to a shift to keep the average utilization of an effective car under
1 in every hour.

NEW-DATA
A permanent file which is created by the WRITE command from all or
part of CURRENT-DATA.

NON-CFS WORK
1. Any activity of a patrol car that makes the car unavailable for
dispatch but was not generated by a previous dispatch to a call
for service.
2. Number of car-hours spent on such activities.

OBJECTIVE FUNCTION
The performance measure to be minimized by an allocation.

OPTIMAL
Yielding the smallest possible value of the objective function.

OUTPUT ORDER
A choice of displaying output tables either by tour within day within
precinct, or by precinct within tour within day.

OVERLAY TOUR
A tour that begins during one tour and ends during the following tour.

PARAMETER
A number that characterizes a particular hour, block, shift, day, or
precinct. See also SERVICE TIME PARAMETER and CALL RATE PARAMETER.

PATROL CAR
A mobile vehicle that can respond to calls for service from the
public. Includes vehicles other than automobiles that serve the
same function, e.g., scooters.

PATROL FREQUENCY
Number of times per hour that a random point will be passed by a car
on preventive patrol.

PCAM
Patrol Car Allocation Model.
PLUS (+)
At the start of a line of output from the DISP command, indicates that the tour is an overlay.

POISSON PROCESS
In the PCAM context, the occurrence of calls for service in a given precinct during a given hour constitutes a Poisson process if there is a parameter $\lambda$ such that the time between calls has the distribution

$$\text{Prob}(\text{time between calls} > t) = e^{-\lambda t}.$$ 

This assumption is well verified by data.

PRECINCT
A geographical area that is treated as independent from other areas by the patrol car dispatcher. Each patrol car is assigned to an entire tour in one precinct, although it may work in only part of the precinct.

PRESCRIPTIVE MODE
Capability to suggest the number of patrol cars that should be on duty during each shift, so as to meet standards of performance specified by the user.

PREVENTIVE PATROL
The practice of driving a patrol car through an area, with no particular destination in mind, looking for criminal incidents or opportunities, suspicious occurrences, etc.

PRIORITY
Importance of a call for service. PCAM permits three priority levels. Priority 1 calls are so important that the dispatcher will violate ordinary dispatching practices to get a patrol car to respond immediately. Priority 2 calls are important enough that a rapid response is preferred over a slow response. Priority 3 calls can wait in queue without deleterious effect.

QUALIFIER
Phrase(s) associated with a computer command, defining the scope of the command. May be any subset of these phrases, separated by delimiters: 'TOUR=〈NAMELIST〉', 'DAY=〈NAMELIST〉', 'DIVISION=〈NAMELIST〉', 'PRECINCT=〈NAMELIST〉'.

QUEUE
In the PCAM context, a collection of calls for service that are waiting to be assigned to a patrol car because no patrol car is available at the moment.

QUEUING DELAY
The length of time a call for service waits in queue.
REGRESSION ANALYSIS
A procedure for fitting a straight line to data so as to minimize the sum of the squares of the deviations of the data from the straight-line estimate.

RESPONSE TIME, TOTAL
Sum of queuing delay and travel time. (Same as TOTAL DELAY.)

SCOPE
The collection of precincts, tours, and days to which the action of a PCAM command applies.

SERVICE TIME
Number of minutes a patrol car will be unavailable from the time it is dispatched to a call until it is available to respond to another call.

SERVICE TIME PARAMETER
A parameter for each day in each precinct. When multiplied by the hourly service time factors, gives the expected service time in each hour.

SHIFT
A particular tour in a particular precinct on a particular day.

SQUARE-ROOT LAW
An equation for the average travel distance D in a region of area A when N patrol units are available:

\[
D = \sqrt{\frac{A}{N}}
\]

STEADY STATE
In the PCAM context, a situation where the probability of finding n cars available does not change over the course of an hour.

SUPPRESSIBLE CRIMES
Any crimes whose occurrence might conceivably be affected by the amount of preventive patrol. (It is not known whether any crimes are actually "suppressed" by preventive patrol.) The PCAM user is free to define this category of crimes however he chooses.

TIME BLOCK
See BLOCK, TIME.

TOTAL DELAY
Same as TOTAL RESPONSE TIME; the sum of queuing delay plus travel time.

TOUR
A period of time (whole number of hours) beginning when a patrol officer starts work for the day and ending when the officer finishes work. In PCAM, tours are assumed to start at the same time in every precinct on every day (but overlay tours need not be present on every day in every precinct).
TRAVEL TIME
The length of time from the moment a patrol car is dispatched to an incident until the moment it arrives at the scene.

UNAVAILABILITY PARAMETERS
A pair of constants B1 and B2 for each precinct that give the best regression fit to the linear equation

\[
\left( \text{fraction of time on non-cfs work} \right) = B1 \times \left( \text{fraction of time on cfs work} \right) + B2.
\]

UTILIZATION
The fraction of time a patrol car is busy on cfs work.
I. PROGRAM INSTALLATION

INTRODUCTION

The Patrol Car Allocation Model (PCAM) is a computer program designed to help police departments determine the number of patrol cars to have on duty in each of their geographical commands. Typically, the number of patrol cars needed will vary according to the season of the year, day of the week, and hour of the day.

A companion user's manual describes applications of the program, explains the meaning of the various items of data to be included in the data base, and gives complete instructions for operating the program once it is installed:


The PCAM program is written in the FORTRAN language and is compatible with most FORTRAN compilers. The particular compiler features required by the program will be described below.

Successful use of the PCAM program requires little or no expertise in the use of computers. The user controls the program with a sequence of simple commands. These can be punched on cards for operation in batch mode (where the program's output is produced on a line printer), or they can be entered at a teletype or other slow-speed terminal for operation in interactive mode (in which case the program's output is displayed immediately at the terminal). Some of the facilities provided by the commands are:

- Data selection
- Allocation of patrol cars to meet constraints on performance measures
- Allocation of patrol cars to best achieve specified objectives
- Display of measures describing expected patrol car performance under particular allocations.
The data required for processing these commands must be supplied to the program in an external file that we call DATABASE. The format for this file is described in Chapter II.

Installing PCAM on a computer system is a simple and straightforward operation. However, various computer systems differ with respect to their conventions for accessing files in the FORTRAN language, and this may have to be taken into account in program installation. In addition, users may wish to optimize the amount of run-time storage reserved, with respect to the size of their data base and intended use of the program.

The program as listed in Chapter V and distributed by Rand is set up to run in batch mode, with changes for interactive mode indicated (see "Minor Program Modifications," below). However, on request we will supply the program in a form suitable for interactive operation. If the program is to be used primarily in batch mode, many users will wish to make program changes to enhance the appearance of the output.

This chapter provides the information needed to install the program and make the indicated types of changes. The user wishing to make more substantial changes will have to familiarize himself with Chapters IV and V. Refer to the Glossary and the User's Manual for definitions of unfamiliar terms.

PCAM SOURCE LANGUAGE AND COMPILATION

The PCAM program is written in the FORTRAN language. The program conforms closely to ANSI standards, but two extensions were used to simplify coding. These are:

- Generalized subscripts (subscripts can be coded as arbitrary integer-valued expressions), e.g.,
  IP=ICDAT(IPARM+(IPARM-1)*2+LBLKTE(2))
- Use of quoted literals in FORMAT statements, e.g.,
  1 FORMAT(' THIS FORMAT STATEMENT IS ALLOWED').

When all desired modifications to the source code have been made,

the PCAM program should be compiled, and the object program saved in an execution-ready form. On IBM 360 or 370 systems running under OS or similar operating systems, the following JCL might be used to accomplish this:

```
// jobcard
//STEP1 EXEC FORTGCL
//FORT.SYSTN DD *
:
PCAM source program
:
/*
//LKED.SYSLMOD DD definition of load module library
//LKED.SYSTIN DD *
NAME PCAM
*/
```

**FILE STRUCTURE AND CONVENTIONS**

The basic inputs to the PCAM program are (1) a sequence of commands, supplied by the user on cards or through an interactive terminal, which control the functions performed by the program, and (2) the DATABASE file on a direct access or magnetic tape device, which describes the characteristics of a city that are relevant to PCAM's modeling of its police patrol operations. PCAM's basic operations can only be performed on that part of DATABASE which resides in the computer's main memory. The user directs part or all of the data to be read from DATABASE by means of a READ command, as described in the User's Manual. The term CURRENT-DATA refers to the data that have been read from DATABASE and are available for processing.

The user can modify the contents of CURRENT-DATA through various commands. PCAM also has the capability of writing out a file containing part or all of the information in CURRENT-DATA. The file created by this operation is called NEW-DATA and is written in response to a WRITE command (see the User's Manual). It is in the same format as DATABASE and can be used in its place in subsequent runs of PCAM.

PCAM references all files through INTEGER variables that contain FORTRAN unit numbers. This facilitates changing the unit numbers used by PCAM to conform to the conventions of a particular operating system.
Table 1 describes PCAM's files in terms of these file reference variables and gives the values of the variables in the distributed program. All file reference variables (except the variable for NEW-DATA) are in COMMON/SYSTEM/. To change the values of these file reference variables, the DATA statement numbered 3870 in Chapter V, located in the BLOCK DATA subprogram, should be modified.

### Table 1

**PCAM FILES**

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Device</th>
<th>Description</th>
<th>Unit Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSIN</td>
<td>Teletype, terminal, card reader, disk, tape, etc.</td>
<td>User's command input</td>
<td>4</td>
</tr>
<tr>
<td>SYSOUT</td>
<td>Teletype, terminal printer, disk, tape, etc.</td>
<td>Printed output from program</td>
<td>5</td>
</tr>
<tr>
<td>IFILE</td>
<td>Tape, direct access</td>
<td>Input data (DATABASE)</td>
<td>19</td>
</tr>
<tr>
<td>NUNIT</td>
<td>Tape, direct access</td>
<td>Output data base (NEW-DATA)</td>
<td>Value determined from user input in the WRITE command</td>
</tr>
<tr>
<td>LIT</td>
<td>Tape, direct access</td>
<td>Scratch file for literals (space needed for three 80-character records)</td>
<td>20</td>
</tr>
</tbody>
</table>

Whether the user changes the unit numbers or not, most operating systems require the user to prepare data definition statements that identify the device or file corresponding to each unit number. For example, the following JCL is used to run the program on the Rand Computation Center's IBM 370 computer:
The FT18 DD statement allows the user to write a NEW-DATA file on FORTRAN unit 18. In other words, the user is permitted to enter the command WRITE [DATA] [ON] 18 [FOR] <QUALIFIER>.

STORAGE ALLOCATION

Most run-time storage that PCAM uses is allocated dynamically from a large one-dimensional array (see Chapter IV). The array is named CDAT (an abbreviation for CURRENT-DATA) and is contained in COMMON/STORE/. Wherever COMMON/STORE/ occurs in the program, an array ICDAT of the same size as CDAT is defined and is equivalenced to CDAT. The minimum amount of storage that must be reserved for CDAT depends on the size of the user's data base and on how much of the data base will be accessed in a single READ command.

Four different classes of information are stored in CDAT. The storage requirements for each class are given below. The sum of these requirements is the minimum size for array CDAT.

1. Permanent Tables

Tables that are allocated at the start of each PCAM run require the following number of words of storage:

\[(9 \cdot \text{NDAYDT}) + (13 \cdot \text{NTRDT}) + (3 \cdot \text{NBLDT}) + (8 \cdot \text{NDIVDT}),\]

where \(\text{NDAYDT} = \text{number of days of data in DATABASE},\) \(\text{NTRDT} = \text{number of tours in each day in DATABASE},\) \(\text{NBLDT} = \text{number of blocks for each day in DATABASE},\) and \(\text{NDIVDT} = \text{number of divisions in DATABASE}.)
2. Variable Size Tables

Tables whose size depends on the number of divisions, days, and tours selected in a READ command qualifier require the following number of words of storage:

\[ \text{NDAYRD} + \text{NTRRD} + \text{NDIVRD}, \]

where \( \text{NDAYRD} \) = number of days read into CURRENT-DATA, \( \text{NTRRD} \) = number of tours read, and \( \text{NDIVRD} \) = number of divisions read.

3. Data Storage

Data read from DATABASE into CURRENT-DATA by a READ command require the following number of words of storage:

\[ \text{NPCTRL} \cdot \text{NWDPCT}, \]

where \( \text{NPCTRL} \) = number of precincts included in CURRENT-DATA and the calculation of \( \text{NWDPCT} \) will be explained below in Chapter IV (Table 7).

4. Temporary Storage

Temporary storage is used for names and numbers* during command interpretation and execution. The exact amount of this storage that is needed for any command is given by:

\[ 8 \cdot (\text{number of names in command}) + 2 \cdot (\text{number of numbers in command}) \]

Temporary storage is always released when the execution of a command is complete. An allocation of about 150 words for this type of storage will be sufficient for most applications.

The program as distributed allows 11,000 words for the sum of these four requirements. Most users will find this amount of space adequate. If the user's DATABASE is too large, an error message will be printed when the program is run, and then it will be necessary to calculate the actual requirements as listed above.

---

*Names appear in user commands to label entities such as precincts, days, and tours. Numbers are used to identify output tables, objective functions, constraints, files, and numerical quantities. For more detailed information, see Chapter II and also the User's Manual.
If the user's DATABASE requires less than 11,000 words for CDAT, the program will operate properly, and a message will be printed after the END command indicating the total number of words actually used for the first three requirements listed above. The user can then reduce the amount of space allocated to CDAT, if he desires to do so. The only advantage in making this modification will be a reduction in the cost of running the program.

In order to change the space allocation, either an increase when necessary or a decrease when desired, the dimensions of CDAT and ICDAT must be changed on the following pairs of lines of the program (see Chapter V):

| 13, 15 | 2210, 2212 |
| 81, 83 | 2263, 2265 |
| 215, 217 | 2311, 2313 |
| 260, 262 | 2365, 2367 |
| 565, 567 | 2419, 2421 |
| 618, 620 | 2573, 2575 |
| 996, 998 | 2749, 2751 |
| 1153, 1155 | 2875, 2877 |
| 1203, 1205 | 2953, 2955 |
| 1232, 1234 | 2976, 2978 |
| 1365, 1367 | 3242, 3244 |
| 1450, 1452 | 3285, 3287 |
| 1493, 1495 | 3322, 3324 |
| 1532, 1534 | 3402, 3404 |
| 1586, 1588 | 3444, 3446 |
| 1685, 1687 | 3486, 3488 |
| 1775, 1777 | 3623, 3625 |
| 1947, 1949 | 3862, 3864 |

In addition, in BLOCK DATA, NWORDS must be set equal to the dimension of CDAT. This occurs on line 3866.

CORE REQUIREMENTS

A moderate amount of core is required to run the PCAM program. Although the exact amount will vary from one computer system to another, the figures given below will serve as a good guideline.

The core requirements for the PCAM program depend directly on the size of the array CDAT; call this NWORDS. The amount of core
required to run PCAM on the Rand Computation Center IBM System 370 computer is given by: \((115 + 4 \cdot \text{NWORDS}/1024)\) K bytes. The program as distributed has \(\text{NWORDS}=11,000\), and therefore requires 158K bytes of core, but we suggest requesting 160K bytes. For installations with other than IBM 360/370 computers, the equivalent requirement can be obtained by using the fact that there are four bytes in a word on IBM 360/370 computers and that 1K byte = 1024 bytes.

For the assistance of potential users who are severely restricted in the amount of core, we point out that the core requirements for PCAM can be reduced by means of a technique called chaining.* (We do not recommend chaining the PCAM program unless absolutely necessary.) This technique allows only those parts of the program that are required to perform a particular function to be resident in core. The remainder of the program can remain in external storage until required. For example, an examination of the program listing in Chapter V and the cross-referenced listing of program segments in Appendix C reveals that while a LIST command is being executed (by subroutine LIST) there is no need for subroutine WRITE, which implements the WRITE command, to be core-resident.

The best way to break up the PCAM program for chaining will vary from installation to installation and depends upon the amount of core available to run the program and the way in which it is used. In general, those subprograms required for the execution of all, or most, commands should be core-resident throughout the program's execution. The subprograms that are needed to execute particular commands should be grouped so that only the group required to execute one command resides in core with the continuously resident subprograms.

MINOR PROGRAM MODIFICATIONS

To convert the PCAM batch program into an interactive program, five lines must be removed from the program (43, 44, 51, 416, and 417), and two lines must be inserted (25 and 26). These are described in Chapter V under the headings "MAIN Program" and "Subroutine GETTKN."

*This is frequently called "overlaying," but we do not use this terminology because the word "overlay" has been assigned a specific (and different) meaning in the User's Manual and the Glossary.
If the program is operated in batch mode, the user may wish to change the appearance of output, since there is more room on a page of output from a high-speed printer than there is on output from a teletype or similar interactive terminal. Changes in spacing of columns of output, number of decimal places displayed, and column labels may be made as follows:

- For the table displayed by the LIST command, change format statements in Subroutine LIST.
- For the tables displayed by the DISP command, change format statements in Subroutine TITLE and Subroutine PRTBL (print table).

In addition, output data of no interest to the user can be completely suppressed by modifying the same subroutines.

To change pagination, or to provide a different heading at the top of the first page (for example, the name of the police department or the date of the run), modify the MAIN program or the INIT subroutine.

In all tours*, the department may prefer to have PCAM allocate cars rather than car-hours. The program as distributed will accept commands referring to cars, such as ALOC 24 CARS BY F(2), but it will interpret the expression "24 cars" to mean 24 car-hours. To change the program so that the input number is interpreted as cars, line 3103 of the program must be changed following the instructions on lines 3098-3102 (see Chapter V).

**COSTS**

The cost of running the PCAM program will vary from installation to installation. However, we can give a rough idea of the range of costs based on our experience with two computer systems. Compiling the program costs approximately $10, and this is more expensive than

---

*In the PCAM documentation, the word "tour" is used to designate the period of time during which a patrol car is on duty. The program itself employs whatever term the user specifies—watch, shift, platoon, or whatever.
most runs of the program after compilation. It is therefore desirable to save the object code from the compiled program.

The demonstration of the program illustrated in the figures in Chapter III of the User's Manual was run from object code at a cost under $2. Typical applications should involve costs under $10 for machine time unless numerous ADD or ALOC commands are entered, or these commands are performed on a large number of shifts simultaneously. (In PCAM terminology, a shift is a tour in all precincts at once. The number of shifts is the product of the number of precincts, days, and tours included explicitly or implicitly in a command qualifier.) In general, PCAM is an inexpensive program to operate and compares favorably with any other program that could answer similar policy questions.
II. PCAM DATA FILE FORMAT

This chapter describes the format of the DATABASE and NEW-DATA files mentioned in Chapter I. Figure 1 and the demonstration DATABASE in Appendix A may assist the user in interpreting the instructions in this chapter. The format items shown are those used to read the DATABASE file. Those used to write NEW-DATA files are different in some respects as noted, but always produce a file that can later be read according to the formats shown for DATABASE. The reader is referred to the Glossary and the User's Manual for the definitions of unfamiliar terms. Appendix B documents a computer program (not part of PCAM) that may assist some departments in calculating values for the unavailability parameters B1 and B2 that appear in the precinct header record, described below.

The DATABASE file must be prepared in standard 80-column records on a disk or other rewindable storage device. The PCAM program uses the variable name IFILE for DATABASE and assumes it is located on unit 19. The user may change the unit number in COMMON/SYSTEM/, which is initialized on line 3870 in BLOCK DATA (see Chapter V).

1. Control record. This is the first record in the data base.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8</td>
<td>A8</td>
<td>Left justify</td>
<td>The word DIVISION, or whatever word the department uses for aggregations of precincts.</td>
</tr>
<tr>
<td>11-18</td>
<td>A8</td>
<td>Left justify</td>
<td>The word PRECINCT, or whatever word the department uses for precincts.</td>
</tr>
<tr>
<td>21-28</td>
<td>A8</td>
<td>Left justify</td>
<td>The word TOUR, or whatever word the department uses for tour.</td>
</tr>
<tr>
<td>30-31</td>
<td>I2</td>
<td>Right justify</td>
<td>Number of divisions in the data base.</td>
</tr>
<tr>
<td>33-35</td>
<td>I3</td>
<td>Right justify</td>
<td>Number of precincts in the data base.</td>
</tr>
<tr>
<td>37-39</td>
<td>I3</td>
<td>Right justify</td>
<td>Number of days of data which are supplied for each precinct.</td>
</tr>
<tr>
<td>41-42</td>
<td>I2</td>
<td>Right justify</td>
<td>Number of time blocks in each day.</td>
</tr>
<tr>
<td>44-45</td>
<td>I2</td>
<td>Right justify</td>
<td>Number of tours in each day.</td>
</tr>
<tr>
<td>47</td>
<td>I1</td>
<td></td>
<td>Indicator for overlay tour. Enter 0 or 1 as described below.</td>
</tr>
</tbody>
</table>

*All A8 formats are read as 8A1.
PCAM permits the following possibilities for overlay tours: (a) there are no overlay tours, (b) every day in every precinct has an overlay tour, or (c) some days and/or some precincts have an overlay tour. Enter 0 for the overlay tour indicator in case (a); enter 1 in case (b) or (c). In cases (b) and (c), the last tour in the data for every day in every precinct must be the overlay tour. However, in case (c) the overlay tour data will be blank for some days and/or precincts.

2. Day name record(s). If there are ten days or fewer in the data base, this is the second record. Otherwise, continuation records will be needed; supply as many as required.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin in 1</td>
<td>10A8</td>
<td>Left justify</td>
<td>Name for each day in the data base. For each precinct, day data will have to be in the same order as the names on this record.</td>
</tr>
</tbody>
</table>

3. Blocks descriptor record. This follows the day name record(s).

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin in 1</td>
<td>24(I2,1X)</td>
<td>Right justify</td>
<td>Last hour of each time block. Supply as many hours as there are time blocks in a day, up to 24. The hours must be in increasing order.</td>
</tr>
</tbody>
</table>

4. Tour descriptor records. If there are ten or fewer days in the data base, and each day has N tours, the tour descriptor records will be the 4th, 5th, ..., 3 + Nth records. There is one such record for each tour, with the overlay tour (if any) last.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8</td>
<td>A8</td>
<td>Left justify</td>
<td>Name of tour.</td>
</tr>
<tr>
<td>10-11</td>
<td>I2</td>
<td>Right justify</td>
<td>Ordinal number of the first time block (or only time block) in the tour.</td>
</tr>
<tr>
<td>13-14</td>
<td>I2</td>
<td>Right justify</td>
<td>Ordinal number of the second time block in the tour. Zero or blank if the tour has only one time block.</td>
</tr>
</tbody>
</table>
5. **Precinct header record.** The first such record follows the last tour descriptor record. The next such record follows all the data for the first precinct. In total, the number of precinct header records will equal the number of precincts in the data.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Comments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8</td>
<td>A8</td>
<td>Left justify</td>
<td>Precinct name.</td>
</tr>
<tr>
<td>10-17</td>
<td>A8</td>
<td>Left justify</td>
<td>Division name for this precinct.</td>
</tr>
<tr>
<td>20-24</td>
<td>F5.0*</td>
<td></td>
<td>Area of precinct (in square miles).</td>
</tr>
<tr>
<td>26-30</td>
<td>F5.0**</td>
<td></td>
<td>Total length of streets in precinct (in miles).</td>
</tr>
<tr>
<td>32-36</td>
<td>F5.0***</td>
<td></td>
<td>Unavailability parameter B1.</td>
</tr>
<tr>
<td>38-42</td>
<td>F5.0***</td>
<td></td>
<td>Unavailability parameter B2.</td>
</tr>
</tbody>
</table>

6. **Day detail records.** There are three of these records for each day in each precinct. The first three follow the first precinct header, the next three appear after all data for shifts and blocks in the first day in the first precinct, etc.

<table>
<thead>
<tr>
<th>Record</th>
<th>Columns</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-5</td>
<td>F5.0</td>
<td>Call rate parameter.</td>
</tr>
<tr>
<td></td>
<td>7-11</td>
<td>F5.0</td>
<td>Service time parameter.</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>I1</td>
<td>An indicator for the presence or absence of an overlay tour for this day for this precinct. Enter 0 if there is no overlay tour, 1 if there is an overlay tour.</td>
</tr>
<tr>
<td>2</td>
<td>1-72</td>
<td>24(F3.2)+</td>
<td>Call rate factors for each hour of the day. The product of one of these factors and the call rate parameter in record 1 should be the number of calls occurring in the precinct in the corresponding hour of the day.</td>
</tr>
<tr>
<td>3</td>
<td>1-72</td>
<td>24(F3.2)+</td>
<td>Service time factors for each hour of the day. The product of one of these factors and the service time parameter in record 1 should be the average service time for calls occurring in the precinct in the corresponding hour of the day.</td>
</tr>
</tbody>
</table>

* F5.2 in NEW-DATA.
** F5.1 in NEW-DATA.
*** F5.3 in NEW-DATA.
+ No decimal points in NEW-DATA.
7. **Shift detail records.** There is one such record for each tour for each day for each precinct. After the third day detail record for each day in each precinct, there will be N of these records, describing the N tours in that day.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>F5.0*</td>
<td>Average number of cars on duty during the shift.</td>
</tr>
<tr>
<td>7-11</td>
<td>F5.0*</td>
<td>Average speed of cars when responding to calls (in miles/hour).</td>
</tr>
<tr>
<td>13-17</td>
<td>F5.0*</td>
<td>Average speed of cars when on preventive patrol (in miles/hour).</td>
</tr>
<tr>
<td>19-23</td>
<td>F5.0**</td>
<td>Fraction of calls which are of priority 1.</td>
</tr>
<tr>
<td>25-29</td>
<td>F5.0**</td>
<td>Fraction of calls which are of priority 2.</td>
</tr>
</tbody>
</table>

8. **Block detail records.** There is one such record for each day for each precinct. It must follow the shift detail records for that day and precinct. The number and order of the entries in this record must be the same as in the blocks descriptor record.

<table>
<thead>
<tr>
<th>Columns</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin in</td>
<td>24F3.1</td>
<td>Average total number of suppressible crimes (or outside robberies, etc.) occurring in each time block. These may not be zero.</td>
</tr>
</tbody>
</table>

\*F5.1 in NEW-DATA.

\**F5.4 in NEW-DATA.
Fig. 1 — Order of data in DATABASE
III. ALGORITHMS FOR CONVERSION OF ALLOCATIONS BETWEEN TOURS AND BLOCKS

This chapter documents the algorithms that PCAM uses to convert allocations of cars to tours into block allocations, and vice versa. Recall that a tour is a period of time over which a patrol car can be on duty, and a block is a part of a tour during which the number of patrol cars on duty is constant.

CONVERSION OF TOUR ALLOCATIONS TO BLOCK ALLOCATIONS

Given an allocation of cars to the tours of a day in a precinct, PCAM determines the resulting allocation of cars to blocks as follows:

1. Set the number of cars assigned to each block of the day to zero.
2. For each tour of the day, including the overlay tour if any, add the number of cars assigned the tour to the number of cars assigned to each of its blocks. For example, consider four blocks named 1, 2, 3, and 4, and three tours named A, B, and C. Tour A works blocks 1 and 2, tour B works blocks 2 and 3, and tour C works blocks 3 and 4. Let \( N_1 \) be the number of cars on duty during block 1, and let \( M_a \) be the number of cars assigned to tour a. Then \( N_0 = M_A \), \( N_2 = M_A + M_B \), \( N_3 = M_B + M_C \), \( N_4 = M_C \).

CONVERSION OF BLOCK ALLOCATIONS TO TOUR ALLOCATIONS

PCAM uses two different algorithms to convert block allocations to tour allocations. One algorithm is used to determine the allocation of cars to tours after constraints have been met for blocks. The other algorithm is used to determine the required increase in car allocation when the number of cars allocated to tours is not enough cars to handle the call-for-service workload in all hours of a day.
1. The algorithm to determine the tour allocation after constraints are met for all blocks of a day in a precinct is the following. For each tour of the day:
   a. If the tour is not involved in an overlay, assign the maximum of the number of cars in its blocks.
   b. If an overlay tour starts during the tour, assign the number of cars assigned to its first block. Save this as \( N_1 \). Save the number of cars assigned to its second block as \( N_2 \).
   c. If an overlay tour ends during the tour, assign the number of cars assigned to its second block. Save this as \( N_4 \). Save the number of cars assigned to its first block as \( N_3 \).
   d. If the tour is an overlay tour, assign \( N = \max(N_2 - N_1, N_3 - N_4, 0) \) cars.
   e. If \( N \) equals 0, STOP.
   f. Let \( \delta = \max(N_2 - N_1, 0) - \max(N_3 - N_4, 0) \).
   g. If \( \delta > 0 \) and the overlay tour is longer than the first overlaid tour, add \( \delta \) cars to the first overlaid tour and remove \( \delta \) cars from the overlay tour. If \( \delta < 0 \) and the overlay tour is longer than the second overlaid tour, add \( \delta \) cars to the second overlaid tour and remove \( \delta \) cars from the overlay tour. If \( \delta = 0 \) or the overlay tour is not longer than the overlaid tour, make no adjustments.

2. Under conditions where the allocation of cars to the blocks of a day is insufficient to handle the call-for-service workload in each hour (this can result from the execution of a READ or SET command), the following algorithm is used to determine where to increase the assignment of cars.
   a. Determine the minimum number of cars required for each deficient block and assign that number of cars. Mark such blocks as deficient.
   b. For each tour of the day (the overlay tour last):
      i. If the tour is not involved in an overlay and has a deficient block, assign the maximum number of cars assigned to its blocks.
ii. If an overlay tour starts during the tour, save the number of cars assigned to its first block as \( N_1 \). If its first block is deficient, assign \( N_1 \) cars to the tour.

iii. If an overlay tour ends during the tour, save the number of cars assigned to its second block as \( N_4 \). If its second block is deficient, assign \( N_4 \) cars to the tour.

iv. If the tour is an overlay and one or both of its blocks is deficient, assign cars as follows. Let \( N_2 \) be the number of cars assigned to its first block and \( N_3 \) be the number of cars assigned to its second block (these include the effect of increasing a block assignment to meet the workload restriction in step a above. However, note that at this stage in the algorithm, any assignments made to tours in steps b and c have not been converted to block assignments). The number of cars assigned to the overlay tour is then \( \max(N_2 - N_1, N_3 - N_4) \), number cars currently assigned).

c. The algorithm described above (Conversion of Tour Allocations to Block Allocations) is then used to redetermine the block assignments.
Chapter IV
INTERNAL DATA STRUCTURES

This chapter describes PCAM's internal data structures and its run-time storage management system. An understanding of these aspects of the program is necessary only if the user wishes to interpret the program listings in Chapter V or modify the program. This section assumes a familiarity with the data base format given in Chapter II. Refer to the Glossary and the User's Manual for definitions of unfamiliar terms.

STORAGE MANAGEMENT

The amount of core storage required by PCAM will vary according to the size of the data base and the portion of the data base selected in each READ command. To allow for this variation, while enabling the program to run with the minimum amount of core storage required for a particular data base, PCAM dynamically allocates much of the storage that it uses.

Dynamic storage allocation is accomplished by reserving a large, one-dimensional array, the size of which can be set when the program is compiled. Then, when a variable amount of storage is required for some purpose, it can be allocated from the array, at which time the subscript of the first word allocated is saved for future reference. The array is referenced by the variable name CDAT when REAL data are accessed and by ICDAT when INTEGER data are accessed.

Two subroutines are used to allocate storage from CDAT; these are GETBOT and GETTOP. GETBOT allocates storage from the "bottom" of the array. This storage is used for three types of data: (a) tables whose dimensions depend only on certain parameters describing the data base and do not vary during program execution; (b) tables whose dimensions can change as a result of the number of days and tours selected in a READ command, and (c) data read from the data base by a READ command. Subroutine GETTOP allocates storage from the "top" of CDAT. This is basically "scratch pad" storage, used during interpretation of all commands and sometimes during command execution.
The storage management system is actually rather simple. Storage is allocated on a last-in-first-out basis. Each routine that requests storage has the responsibility of releasing it or not, depending upon intended future use. Storage is released by setting a pointer to a subscript which represents the highest or lowest free word of CDAT, depending upon whether storage is being freed from the top or bottom. Thus, care has been exercised so that storage is not prematurely freed and unrecoverable "holes" are not left in allocated storage.

**TABLE POINTERS**

This section describes the dynamically allocated tables used by PCAM. Pointers to these tables and table dimensions are saved in COMMON/PNTRS/. This common block also contains certain variables relating to overlay tours. For completeness, these variables will also be described in this section. Table 2 below lists each variable in COMMON/PNTRS/, its contents, and the routine where its value is set. If storage is allocated for a table in a routine other than the one in which its entries are made, the name of the routine making the table entries appears in parentheses. Variables beginning with 'N' are dimensions or counters, while those beginning with 'L' are pointers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
<th>Where set (entered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPCTDT</td>
<td>Number of precincts in the database.</td>
<td>INIT</td>
</tr>
<tr>
<td>NPCTRD</td>
<td>Number of precincts read by the last READ command.</td>
<td>READ</td>
</tr>
<tr>
<td>LPCTDT</td>
<td>Pointer to (subscript in CDAT of) data read by last READ command.</td>
<td>READ</td>
</tr>
<tr>
<td>LNMLST(1)</td>
<td>Pointer to list of day names in current command qualifier (stored one character to a word, eight characters to a name).</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>Name</td>
<td>Contents</td>
<td>Where set (entered)</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>LNMLST(2)</td>
<td>Pointer to list of tour names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>LNMLST(3)</td>
<td>Pointer to list of division names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>LNMLST(4)</td>
<td>Pointer to list of precinct names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>NNAMES(1)</td>
<td>Number of day names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>NNAMES(2)</td>
<td>Number of tour names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>NNAMES(3)</td>
<td>Number of division names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>NNAMES(4)</td>
<td>Number of precinct names in current command qualifier.</td>
<td>GTDSPC</td>
</tr>
<tr>
<td>NDAYDT</td>
<td>Number of days of data in the data base for each precinct.</td>
<td>INIT</td>
</tr>
<tr>
<td>LDAYNM</td>
<td>Pointer to table of names of all days in the data base (8*NDAYDT words). These are in the same order as the day data for each precinct in the data base.</td>
<td>INIT</td>
</tr>
<tr>
<td>LDYRFL</td>
<td>Pointer to table of day &quot;read&quot; flags (NDAYDT words). Each entry corresponds to one day in the data base. An entry value of zero indicates that no data are to be read for that day. A nonzero value indicates that data are to be read. If the value is nonzero, then it is the ordinal position of that day among days read. If there are three days' data for each precinct in the data base, and the user selects the first and third in a READ command, then the entries in this table will be 1, 0, 2.</td>
<td>INIT(READ)</td>
</tr>
<tr>
<td>NDAYRD</td>
<td>Number of days of data selected in the last READ command qualifier.</td>
<td>READ</td>
</tr>
</tbody>
</table>
Table 2--continued

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
<th>Where set (entered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDYWFL</td>
<td>Pointer to table of day &quot;work&quot; flags (NDAYRD words). Each entry corresponds to one day for which data have been read. An entry value of zero indicates that the current command will not operate on data for that day. A nonzero value indicates that the day is to be included in the command scope. If the entry is nonzero, then it is the ordinal position of the selected day among all days in the data base. Continuing the above example, if the user selects the second of the days read in a command, then the entries in this table would be 0, 3.</td>
<td>READ(SETWFL)</td>
</tr>
<tr>
<td>NTRDT</td>
<td>Number of tours in the data base for each day.</td>
<td>INIT</td>
</tr>
<tr>
<td>LTRTB(1)</td>
<td>Pointer to table of blocks (NTRDT words). Each entry corresponds to a tour, the order being the same as in the data base. Entry values are the ordinal position among blocks of the first block in a tour.</td>
<td>INIT</td>
</tr>
<tr>
<td>LTRTB(2)</td>
<td>The same as LTRTB(1), except gives the position of the second block for each tour. A zero-valued entry indicates that there is no second block for the tour.</td>
<td>INIT</td>
</tr>
<tr>
<td>LTRST</td>
<td>Pointer to starting hours of tours (NTRDT words). Each entry corresponds to a tour. The value of each entry is the starting hour (1-24) of that tour.</td>
<td>INIT</td>
</tr>
<tr>
<td>LTRTEND</td>
<td>The same as LTRST, but ending hours.</td>
<td>INIT</td>
</tr>
<tr>
<td>LTRRFL</td>
<td>Pointer to table of tour &quot;read&quot; flags. This is the same as LDYRFL, but for tours.</td>
<td>INIT(READ)</td>
</tr>
<tr>
<td>LTRNM</td>
<td>Pointer to table of tour names (8*NTRDT words). These are in the same order as the tour data for each day in the data base.</td>
<td>INIT</td>
</tr>
</tbody>
</table>
Table 2—continued

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
<th>Where set (entered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NTRRD</td>
<td>Number of tours selected in the last READ command qualifier.</td>
<td>READ</td>
</tr>
<tr>
<td>LTRWFL</td>
<td>Pointer to table of tour &quot;work&quot; flags (NTRRD words). This is the same as LDYWFL, but for tours.</td>
<td>READ(SETWFL)</td>
</tr>
<tr>
<td>NBLDT</td>
<td>Number of blocks for each day in the data base.</td>
<td>INIT</td>
</tr>
<tr>
<td>LBLKT(1)</td>
<td>Pointer to table of starting hours for each block (NBLDT words).</td>
<td>INIT</td>
</tr>
<tr>
<td>LBLKT(2)</td>
<td>Pointer to table of ending hours for each block (NBLDT words).</td>
<td>INIT</td>
</tr>
<tr>
<td>LBLRFL</td>
<td>Pointer to table of block &quot;read&quot; flags (NBLDT words). This is the same as LTRRFL, but for blocks.</td>
<td>INIT(READ)</td>
</tr>
<tr>
<td>NBLRD</td>
<td>Number of blocks read by a READ command for each day (function of the number of tours selected).</td>
<td>READ</td>
</tr>
<tr>
<td>LBLWFL</td>
<td>Not used.</td>
<td></td>
</tr>
<tr>
<td>NDIVDT</td>
<td>Number of divisions into which precincts are aggregated.</td>
<td>INIT</td>
</tr>
<tr>
<td>NDIVRD</td>
<td>Number of divisions selected by a READ command.</td>
<td>READ</td>
</tr>
<tr>
<td>LDIVVM</td>
<td>Pointer to list of names of divisions selected by a READ command (8*NDIVDT words). Includes those selected by a request for all precincts.</td>
<td>INIT(READ)</td>
</tr>
<tr>
<td>LDIVFL</td>
<td>Pointer to list of flags that select divisions for current command (not READ) (NDIVRD words). Each entry corresponds to a division name in LDIVVM. A nonzero entry value indicates that the division was selected; a zero entry indicates that it was not.</td>
<td>READ(SETWFL)</td>
</tr>
</tbody>
</table>
Table 2--continued

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
<th>Where set (entered)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOVRLY</td>
<td>A flag that indicates whether or not there are overlay tours in the data base. A value of 1 indicates that the last tour of each day in the data base is an overlay tour; a value of 0 indicates that there are no overlay tours.</td>
<td>INIT</td>
</tr>
<tr>
<td>IOVTR(1)</td>
<td>The position of the first overlaid tour among the tours specified in a READ command. A value of n indicates that the n-th tour of the tours read for each day is the tour during which the overlay tour starts.</td>
<td>READ</td>
</tr>
<tr>
<td>IOVTR(2)</td>
<td>The position of the second overlaid tour among the tours specified in a READ command. A value of m indicates that the m-th tour of the tours read for each day is the tour during which the overlay tour ends (of course, IOVTR(2)=IOVTR(1)+1).</td>
<td>READ</td>
</tr>
</tbody>
</table>

DATA STORAGE

PCAM stores the data read by a READ command in array CDAT in a structure parallel to the way the data are stored in DATABASE (see Chapter II and in particular Fig. 1). For each precinct, a constant-size area of storage contains certain data that describe the precinct as a whole; then a variable-size area contains data for each day read for the precinct. Within the area allocated for each day, a constant-size area contains data for the day as a whole and two variable-size areas contain data for each tour and each block (the size of each area depends on the number of tours read for a day).

Each element of precinct, day, tour, and block data is referenced by a pointer which is the subscript within CDAT of the data for the precinct, day, tour, or block, plus an offset that corresponds to the type of data being referenced. For example, the word containing the area of a precinct is referenced in the program by CDAT(LPCT+ARPOFF), where LPCT is the previously determined pointer to the data for the
precinct and ARPOFF is the relative position within precinct data (for all precincts) of the word containing the precinct's area. Tables 3, 4, 5, and 6 give the layout of the constant data for precincts, days, tours, and blocks. LPCT, LDAY, LTOUR, and LBLK are pointers to particular precincts, days, tours, and blocks, respectively.

Table 3

DESCRIPTION OF PRECINCT DATA

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Mode</th>
<th>Reference</th>
<th>Offset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (8 words)</td>
<td>Character</td>
<td>(LPCT+NMPOFF)</td>
<td>0</td>
</tr>
<tr>
<td>Division number (relative position in LDIVNM of division name: 0 for none)</td>
<td>Integer</td>
<td>(LPCT+DVPOFF)</td>
<td>8</td>
</tr>
<tr>
<td>Area (square miles)</td>
<td>Real</td>
<td>(LPCT+ARPOFF)</td>
<td>9</td>
</tr>
<tr>
<td>Total street length (miles)</td>
<td>Real</td>
<td>(LPCT+SMPOFF)</td>
<td>10</td>
</tr>
<tr>
<td>Bl (unavailability parameter)</td>
<td>Real</td>
<td>(LPCT+B1POFF)</td>
<td>11</td>
</tr>
<tr>
<td>B2 (unavailability parameter)</td>
<td>Real</td>
<td>(LPCT+B2POFF)</td>
<td>12</td>
</tr>
<tr>
<td>Data for days</td>
<td>(a)</td>
<td>(LPCT+DYPOFF)</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 4

DESCRIPTION OF DAY DATA

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Mode</th>
<th>Reference</th>
<th>Offset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call rate parameter</td>
<td>Real</td>
<td>(LDAY+CPDOFF)</td>
<td>0</td>
</tr>
<tr>
<td>Service time parameter</td>
<td>Real</td>
<td>(LDAY+SPDOFF)</td>
<td>1</td>
</tr>
<tr>
<td>Overlay indicator for day</td>
<td>Integer</td>
<td>(LDAY+OVDOFF)</td>
<td>2</td>
</tr>
<tr>
<td>Hourly call rates (24 words)</td>
<td>Real</td>
<td>(LDAY+CRDOFF)</td>
<td>3</td>
</tr>
<tr>
<td>Hourly service times (24 words)</td>
<td>Real</td>
<td>(LDAY+STDOFF)</td>
<td>27</td>
</tr>
<tr>
<td>Data for tours</td>
<td>(a)</td>
<td>(LDAY+TRDOFF)</td>
<td>51</td>
</tr>
<tr>
<td>Data for blocks</td>
<td>(a)</td>
<td>(LDAY+BLOFF)</td>
<td>Depends on number of tours</td>
</tr>
</tbody>
</table>

*See Table 5.*
Table 5
DESCRIPTION OF TOUR DATA

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Mode</th>
<th>Reference</th>
<th>Offset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difference in objective function value per car-hour if one car is added to tour</td>
<td>Real</td>
<td>(LTOUR+QDTOFF)</td>
<td>0</td>
</tr>
<tr>
<td>Difference in objective function value per car-hour if a car is removed from an overlay tour and one car is added to each of the tours that it overlays (overlay tours only)</td>
<td>Real</td>
<td>(LTOUR+QXTOFF)</td>
<td>1</td>
</tr>
<tr>
<td>Number of calls during tour</td>
<td>Real</td>
<td>(LTOUR+CRTOFF)</td>
<td>2</td>
</tr>
<tr>
<td>Objective function value with current allocation</td>
<td>Real</td>
<td>(LTOUR+QDTOFF)</td>
<td>3</td>
</tr>
<tr>
<td>Objective function value with one more car</td>
<td>Real</td>
<td>(LTOUR+QXTOFF)</td>
<td>4</td>
</tr>
<tr>
<td>Number of most limiting constraint</td>
<td>Integer</td>
<td>(LTOUR+CTTOFF)</td>
<td>5</td>
</tr>
<tr>
<td>Tour type (1=ignore, 2=standard, 3=first in overlay, 4=second in overlay, 5=overlay tour)</td>
<td>Integer</td>
<td>(LTOUR+TYTOFF)</td>
<td>6</td>
</tr>
<tr>
<td>Actual cars assigned to start tour</td>
<td>Real</td>
<td>(LTOUR+ACTOFF)</td>
<td>7</td>
</tr>
<tr>
<td>Response speed (mph)</td>
<td>Real</td>
<td>(LTOUR+RVTOFF)</td>
<td>8</td>
</tr>
<tr>
<td>Patrol speed (mph)</td>
<td>Real</td>
<td>(LTOUR+PVTOFF)</td>
<td>9</td>
</tr>
<tr>
<td>Fraction of priority 1 calls</td>
<td>Real</td>
<td>(LTOUR+HFTOFF)</td>
<td>10</td>
</tr>
<tr>
<td>Fraction of priority 2 calls</td>
<td>Real</td>
<td>(LTOUR+MFTOFF)</td>
<td>11</td>
</tr>
<tr>
<td>Fraction of priority 3 calls</td>
<td>Real</td>
<td>(LTOUR+LFTOFF)</td>
<td>12</td>
</tr>
</tbody>
</table>
Table 6
DESCRIPTION OF BLOCK DATA

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Mode</th>
<th>Reference</th>
<th>Offset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effective cars (including overlay effects)</td>
<td>Real</td>
<td>(LBLK+EFBOFF)</td>
<td>0</td>
</tr>
<tr>
<td>Actual cars on duty (including overlays)</td>
<td>Real</td>
<td>(LBLK+ACBOFF)</td>
<td>1</td>
</tr>
<tr>
<td>Average workload during hours of block (hours of servicing calls per hour)</td>
<td>Real</td>
<td>(LBLK+AWBOFF)</td>
<td>2</td>
</tr>
<tr>
<td>Total calls during block</td>
<td>Real</td>
<td>(LBLK+CRBOFF)</td>
<td>3</td>
</tr>
<tr>
<td>Maximum workload over all hours of block</td>
<td>Real</td>
<td>(LBLK+RMBOFF)</td>
<td>4</td>
</tr>
<tr>
<td>Number of suppressible crimes during block</td>
<td>Real</td>
<td>(LBLK+OCBOFF)</td>
<td>5</td>
</tr>
<tr>
<td>Number of most limiting constraint</td>
<td>Real</td>
<td>(LBLK+CTBOFF)</td>
<td>6</td>
</tr>
<tr>
<td>Objective function value with current allocation</td>
<td>Real</td>
<td>(LBLK+QOBOFF)</td>
<td>7</td>
</tr>
<tr>
<td>Objective function value with one additional car</td>
<td>Real</td>
<td>(LBLK+QNBOFF)</td>
<td>8</td>
</tr>
</tbody>
</table>

The offsets for all data items described above are contained in COMMON/OFFSET/. Other variables in COMMON/OFFSET/ give the storage requirements for precincts, days, tours, and blocks. These are described in Table 7.

Table 7
OTHER CONTENTS OF COMMON/OFFSET/

<table>
<thead>
<tr>
<th>Variable</th>
<th>Contents</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NWDBL</td>
<td>Number of words required for a block</td>
<td>9</td>
</tr>
<tr>
<td>NWDTR</td>
<td>Number of words required for a tour</td>
<td>13</td>
</tr>
<tr>
<td>NWDDY</td>
<td>Number of words required for a day</td>
<td>[51 + NWDTR \times NTRRD + NWDBL \times NBLRD]</td>
</tr>
<tr>
<td>NPRIO</td>
<td>Number of priority classes</td>
<td>3</td>
</tr>
<tr>
<td>NWDPCT</td>
<td>Number of words required for a precinct</td>
<td>[13 + NWDDY \times NDAYRD]</td>
</tr>
</tbody>
</table>
Chapter V

LISTING* AND DESCRIPTION OF THE PCAM FORTRAN PROGRAM

The discussions in this chapter assume the reader's familiarity with the contents of Chapter IV (Internal Data Structures) and the User's Manual. Refer to Appendix C for a cross-reference listing of program segments and common blocks. An alphabetized list of their names was given immediately following the table of contents.

MAIN PROGRAM

The MAIN program primarily controls the execution of the subroutines that implement the various PCAM commands. It operates in a continuous loop, determining which subroutine to call by examining successive command identifiers, until an END command is encountered.

Execution begins with a call to subroutine INIT to initialize permanent tables, etc. Then, if operating in interactive mode, a message prompting for the user's next command is written. Subroutine SCAN is called to obtain the command identifier. If the identifier is valid, the appropriate subroutine is called to complete command interpretation and execution. When command execution is completed, the MAIN program proceeds to the next command.

The listing provided here is for a batch program. To convert to an interactive program, three clearly indicated changes must be made:

1. Remove the comment 'C' on cards 25 and 26. This will cause the program to prompt for the next command. The user may also have to change the dollar sign on line 26, which serves the purpose of leaving the interactive terminal's print head in place.
2. Remove lines 43 and 44. These cause the printer to eject to a new page after displaying tables of output.
3. Remove line 51. This causes page ejection after listing data.

*In order to meet space constraints, the listing has been photographically reduced by 5 percent.
COMMON/KEYWDS/NKYWD,NTYPES,TYPEOFF(4),KEYW(OB,3),WDTYPE(30)
 INTEGER TYPEOFF,WDTYPE
 DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
 EQUIVALENCE (PCLSNM,KEYW(1,4)),(DCLSNM,KEYW(1,3)),
 1(TOURNM,KEYW(1,2))

COMMON/SYSTEM/SYSIN,SYSSUT,IFILE,LIT
 INTEGER SYSIN,SYSSOUT

COMMON/SCODES/SEND,CMD,NMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
 INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
 INTEGER TOP,BOT,RDBOT
 DIMENSION ICDAT(11000)
 EQUIVALENCE (ICDAT,CDAT)

 INTEGER TYPE,VAL
 DIMENSION VAL(2)
 BOT=1
 TOP=NWORDS+1
 CALL INIT
 LGETT=TOP

**** NEXT 2 LINES NEEDED FOR INTERACTIVE MODE ****
 WRITE(SYSOUT,1)
 1 FORMAT/(/ 'COMMAND? ')$
 TYPE=SEND
 CALL SCAN(TYPE,VAL)
 IF(TYPE .EQ. CMD) GO TO 20
 WRITE(SYSOUT,2)
 2 FORMAT(' **INVALID COMMAND - REENTER.')
 TOP=LGETT
 GO TO 10
 ICM=(VAL(1)-TYPEOFF(CMD))
34 GO TO (100,200,300,400,500,600,700,800,900),ICM

 CALL ADDALC(2)
 GO TO 10
 CALL ADDALC(0)
 GO TO 10
 CALL DISP

**** REMOVE NEXT TWO LINES FOR INTERACTIVE MODE ****
 WRITE(SYSOUT,3)
 3 FORMAT(1I1)
 GO TO 10
 WRITE(SYSOUT,4) MAXBOT
 4 FORMAT(' MAXIMUM SIZE OF CURRENT- DATA WAS ',I5,'WORDS')
 STOP
 CALL LIST

**** REMOVE NEXT LINE FOR INTERACTIVE MODE ****
 WRITE(SYSOUT,3)
 GO TO 10
500    CALL MEET
      GO TO 10
600    CALL READ
      GO TO 10
700    CALL SET
      GO TO 10
800    CALL WRITE
      GO TO 10
END
Subroutine INIT

This subroutine performs initialization tasks for PCAM. It is called only once (from MAIN).

The initialization tasks consist primarily of reading control information from the data base and allocating storage for tables whose dimensions will not change during program execution. In addition, starting hours for blocks and starting and ending hours for tours are computed.

The array KEYWD, which contains all command language keywords, is initialized by writing literals on file LIT. (This is a variable name containing a FORTRAN unit number. See Chapter I.) File LIT is read back under A format. This procedure eliminates the need for a DATA statement, which some compilers restrict to initializing only one array element per entry, and simplifies modification of keywords. In addition, the program determines the relative position among keywords (in KEYWD) of the first keyword of each "syntactic type."* The relative positions of the first keywords of each type are saved in array TYPOFF.

*See description of Subroutine SCAN for syntactic types.
SUBROUTINE INIT

SUBROUTINE TO INITIALIZE PERMANENT TABLES, ETC.

COMMON/SYSTEM/SYSIN,SYSDUT,IFILE,LIT
INTEGER SYSIN,SYSDUT

COMMON/PNTRS/IOVRLY,IOVTIR(2),
INPCTDT,NPCTR0,LPCRTOT,LNMLST(4),NNAMES(1),NDAYDT,LDAYNM,
2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTRTREND,LTRRFL,LTRNM,
3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVOT,NDIVRD,
4LDIVNM,LDIVFL
COMMON/KEYWD/2/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WTYPE(30)
INTEGER TYPOFF,WTYPE
DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
(I(TOURNM,KEYWD(1,2))
COMMON/STORE/TOP,BOT,ROBOT,MAXBOT,WORDS,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)

WRITE(SYSDUT,6)

WRITE KEYWORD SCRATCH FILE

WRITE(LIT,11)
WRITE(LIT,12)
WRITE(LIT,13)
REWIND LIT
READ(LIT,10) ((KEYWD(I,J),I=1,8),J=1,NKYWD)

READ CONTROL CARD FROM DATA BASE

REWIND IFILE
READ(IFILE,1) DCLSNM,PCLSNM,TOURNM,NDAYDT,NBLDT,
NTRDT,IOVRLY

ALLOCATE STORAGE SPACE

N=N*NDAYDT
CALL GETBOT(N,LDAYNM)
CALL GETBOT(NDAYDT,LDYRFL)
NL=LDAYNM+N-1
READ(IFILE,2) (CDAT(I),I=LDAYNM,NL)

CALL GETBOT(NBLDT,LBLKTB(1))
CALL GETBOT(NBLDT,LBLKTB(2))
CALL GETBOT(NBLDT,LBLRFL)
K=LBLKTB(1)-1
READ(IFILE,3) (ICDAT(K+I),I=1,NBLDT)
ICDAT(LBLKTB(1))=1
DO 100 I=2,NBLDT
   ICDAT(LBLKTB(1)+I-1)=ICDAT(LBLKTB(2)+I-2)+1
119
120
121
122
N=8*NTROT
CALL GETBOT(N,LTRNM)
123
124
CALL GETBOT(NTROT,LTRTB(1))
CALL GETBOT(NTROT,LTRTB(2))
CALL GETBOT(NTROT,LTRRF)
125
126
L2=LTRNM-1
127
DO 120 I=1,NTROT
128
L1=L2+1
129
L2=L1+7
130
J=LTRTB(1)+I-1
131
K=LTRTB(2)+I-1
132
READ(IFILE,4) (ICDAT(L3),L3=L1,L2),ICDAT(J),ICDAT(K)
133
CALL GETBOT(NTROT,LTRST)
134
CALL GETBOT(NTROT,LTREN)
135
136
137
CALCULATE STARTING AND ENDING HOURS
138
139
DO 130 ITOUR=1,NTROT
140
IBLK=ICDAT(LTRTB(1)+ITOUR-1)
141
START=ICDAT(LBLKTB(1)+IBLK-1)
142
ICDAT(LTRST+ITOUR-1)=START
143
IEND=ICDAT(LBLKTB(2)+IBLK-1)
144
ICDAT(LTREN+ITOUR-1)=IEND
145
IF(IBLK .EQ. 0) GO TO 130
146
IEND=ICDAT(LBLKTB(2)+IBLK-1)
147
ICDAT(LTREN+ITOUR-1)=IEND
148
149
CONTINUE
150
151
CALL GETBOT(8*NDIVDT,LDIVMV)
152
ROBOT=BOT
153
154
ASSIGN TYPES TO KEYWORDS
155
156
I=0
DO 150 ITYPE=1,NTYPES
157
N=TYPOFF(ITYPE)
158
TYPOFF(ITYPE)=I
159
DO 140 J=1,N
160
I=I+1
161
IF(I .LT. NKYWD) GO TO 140
162
WRITE(SYSOUT,5) ITYPE
163
STOP
5
CONTINUE
164
STOP
165
REWRITE IFILE
166
RETURN
167
FORFORMAT(8A1,2X,8A1,2X,8A1,IX,2X,8A1,IX,1X,I3,1X,I3,1X,I3,1X,I2,1X,I2,1X,11)
168
FORMAT(80A1)
169
FORMAT(24(I2,1X))
170
FORMAT(8A1,IX,I2,1X,I2)
171
FORMAT(12X,'NEW YORK CITY - RAND INSTITUTE'/27X,172
C 'PATROL CAR ALLOCATION MODEL'//)
173
174
175
176
formatted
C'DAY", 177
C'TOUR", 178
C'DIVISION", 179
C'PRECINCT", 180
C'P", 181
C'C", 182
C'T", 183
C'F", 184
C'ADD", 185
C'ALOC") 186
          ) 187
          ) 188
C'DISP", 189
C'END", 190
C'LIST", 191
C'MEET", 192
C'READ", 193
C'SET", 194
C'WRITE", 195
C'FOR", 196
C'CAR", 197
C'CARS") 198
          ) 199
C'TO", 200
C'BY", 201
C'DATA", 202
C'HOUR", 203
C'HOURS", 204
C'ON") 205
          ) 206
          ) 207
C I2," - EXECUTION TERMINATED") 208
END 209
Subroutine GETBOT

Subroutine GETBOT (get bottom) allocates storage from the "bottom" of array CDAT.* The input parameter N specifies the number of words of storage that are needed. The variables TOP and BOT in COMMON/STORE/ contain the subscripts of the highest free word plus one and the lowest free word in CDAT, respectively. If N words of storage are available, the output parameter L is set to the subscript of the first word allocated, BOT is updated, and the storage obtained is set to zeros. If N words of storage are not available, execution is terminated.

SUBROUTINE GETBOT(N,L)
COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT
COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWJROS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE(ICDAT,CDAT)

ALLOCATE STORAGE

L=BOT
BOT=L+N

ERROR CONDITION . INSUFFICIENT SPACE
IF(BOT .LT. TOP) GO TO 10
WRITE(SYSOUT,1)
1 FORMAT('/ *** INSUFFICIENT STORAGE FOR TABLES OR DATA - ',
1'EXECUTION TERMINATED*)
STOP

SET DATA TO ZERO

K=BOT-1
DO 20 I=L,K
ICDAT(I)=0
20 IF(BOT.GT. MAXBOT) MAXBOT=BOT
RETURN
END

* See Chapter IV for a description of the storage management system.
**Subroutine SCAN**

This subroutine scans the user's command input for the next syntactic element (e.g., command identifier, name list, number list, etc.) Its two parameters STYPE and SVAL are set to the type and value, respectively, of the element obtained. SVAL is a two-word array; the meaning of each word depends on the type of the syntactic element, as shown in Table 8.

<table>
<thead>
<tr>
<th>Type Identifier (STYPE)</th>
<th>Description</th>
<th>Form of SVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEND</td>
<td>End of command encountered</td>
<td>--</td>
</tr>
<tr>
<td>CMD</td>
<td>Command identifier</td>
<td>(Position of identifier in KEYWD table, --)</td>
</tr>
<tr>
<td>NUMLST</td>
<td>Number list</td>
<td>(Number of elements in list, pointer to list)</td>
</tr>
<tr>
<td>NAMLST</td>
<td>Name list</td>
<td>(Number of elements in list, pointer to list)</td>
</tr>
<tr>
<td>FSPEC</td>
<td>Function identifier (objective function, constraint, data type, table)</td>
<td>(Position of identifier in KEYWD table, --)</td>
</tr>
<tr>
<td>DSPEC</td>
<td>Data type specification (DAY, TOUR, PRECINCT, DIVISION)</td>
<td>(Position of identifier in KEYWD table, --)</td>
</tr>
<tr>
<td>ERR</td>
<td>Invalid element</td>
<td>--</td>
</tr>
</tbody>
</table>

**Table 8**

SYNTACTIC TYPES RETURNED FROM SCAN

SCAN calls GETTKN to get the next lexical element (number, word, paren, etc.) from the command text. If STYPE indicates that the last element type was "end of command," then SCAN instructs GETTKN to start reading a new command by setting TYPE to indicate "end of command." (See description of Subroutine GETTKN; the parameter is called LTYPE in SCAN.)

The elements of name lists are stored in the "top" of array CDAT in storage allocated by calls to GETTOP. Names are stored eight characters to a name, one character to a word. Elements of name lists
occupy contiguous words of storage in the order opposite to that in which they were entered.

Numbers are stored in word pairs. The first word of a pair contains the integer representation and the second word the floating point representation of the number. Word pairs in a number list occupy contiguous words of storage in the same order as that in which they were entered.
SUBROUTINE SCAN(STYPE,SVAL)

SCANS USER COMMAND INPUT FOR NEXT LEXICAL ELEMENT

COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT

COMMON/KEYWDS/NKWDS,NTYPES,TYPDFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER TYPDFF,WDTYPE
DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
I(TOURNM,KEYWD(1,2))

COMMON/LCODES/LEND,WORD,NJM,LP,RP
INTEGER WORD,RP

COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

COMMON/STORE/TOP,BOT,ROBOT,MAXBOT,NWJRDS,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)

INTEGER STYPE,SVAL
DIMENSION SVAL(2),LVAL(3)

LGETT=TOP
IF(STYPE .EQ. SEND) LTYPE = LEND

GET NEXT LEXICAL ELEMENT FROM COMMAND

CALL GETTKN(LTYPE,LVAL)
GO TO (100,200,300,400,405),LTYPE

END OF COMMAND REACHED

0 STYPE=SEND
RETURN

BEGINNING OF A WORD ENCOUNTERED

0 I=LKP8(LVAL,KEYWD,NKWDS)
IF(I .EQ. 0) GO TO 220
STYPE=WDTYPE(I)
IF(STYPE .EQ. DUM) GO TO 10
SVAL(I)=I
RETURN

0 STYPE=NAMLST
SVAL(I)=1
CALL GETTOP(8,I)
CALL MOVE(LVAL,CDAT(I),8)
SVAL(2)=I
RETURN

NEXT LEXICAL ELEMENT IS A NUMBER
C 300  STYPE=NUMLST
    SVAL(1)=1
    CALL GETTOP(Z,I)
    ICDAT(I)=LVAL(1)
    ICDAT(I+1)=LVAL(2)
    SVAL(2)=I
    RETURN

C 400  CALL GETTKNL(LTYPE,LVAL)
    IF(LTYPE .EQ. NUM) GO TO 450
    IF(LTYPE .EQ. WORD .OR. LTYPE .EQ. RP) GO TO 410

C 405  WRITE(SYSPUT,1)
    FORMAT(/' *** INVALID LIST FORMAT - REENTER.')
    STYPE=ERR
    RETURN

C 410  N=0
    STYPE=NAMLST.
    IF(LTYPE .EQ. RP) GO TO 430
    IF(LTYPE .EQ. WORD) GO TO 420
    WRITE(SYSPUT,2)
    FORMAT(/' *** INVALID NAME LIST ELEMENT - REENTER.')
    TOP=LGETT
    STYPE=ERR
    RETURN

420  N=N+1
    CALL GETTOP(8,LOC)
    CALL MOVE(LVAL,CDAT(LOC),8)
    CALL GETTKNL(LTYPE,LVAL)
    GO TO 415

430  SVAL(1)=N
    SVAL(2)=LOC
    RETURN

C 450  N=0
    STYPE=NUMLST.
    IF(LTYPE .EQ. RP) GO TO 480
    IF(LTYPE .EQ. NUM) GO TO 470
    WRITE(SYSPUT,3)
    FORMAT(/' *** INVALID NUMBER LIST FORMAT - REENTER.')
    STYPE=ERR
    TOP=LGETT
    RETURN

C 470  N=N+1
    CALL GETTOP(2,LOC)
ICDAT(LOC) = LVAL(1)
ICDAT(LOC+1) = LVAL(2)
CALL GETKBN(LTYPE, LVAL)
GO TO 460
SVAL(1) = N
SVAL(2) = LOC
NSW = N/2
IF(NSW .LT. 1) RETURN
J = LOC
K = LOC+(N-1)*2
DO 490 I = 1, NSW
  IT1 = ICDAT(J)
  IT2 = ICDAT(J+1)
  ICDAT(J) = ICDAT(K)
  ICDAT(J+1) = ICDAT(K+1)
  ICDAT(K) = IT1
  ICDAT(K+1) = IT2
  J = J+2
  K = K-2
CONTINUE
RETURN
END
Subroutine GETTKN

Subroutine GETTKN (get token) obtains the next lexical element in the user's command input. Its two parameters TYPE and VAL are set to the class and value, respectively, of the element obtained. VAL is an eight-word array; its use depends on the type of element scanned, as shown in Table 9.

Table 9

<table>
<thead>
<tr>
<th>Type Identifier (TYPE)</th>
<th>Description</th>
<th>Form of VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEND</td>
<td>End of command</td>
<td>--</td>
</tr>
<tr>
<td>WORD</td>
<td>Character string of up to eight characters, starting with a letter</td>
<td>Each computer word contains one character as if it were read in Al format. WORDs are left adjusted in VAL and padded with blanks</td>
</tr>
<tr>
<td>NUM</td>
<td>Number</td>
<td>The first word of VAL contains the integer representation of the number, and the second word contains its floating point representation.</td>
</tr>
<tr>
<td>LP</td>
<td>Left parenthesis</td>
<td>--</td>
</tr>
<tr>
<td>RP</td>
<td>Right parenthesis</td>
<td>--</td>
</tr>
</tbody>
</table>

At entry, if TYPE indicates that an "end of command" was the last element scanned, a new record is read from the input file and scanned from its start. A new record is also read if an ampersand is encountered while scanning for the next element.

Function LKP1 is invoked to determine the type and value of each character scanned.

Two lines of this subroutine, 416 and 417, must be removed to obtain an interactive program. These print out the command that has just been read, which is unnecessary and annoying if the user has typed the command into his terminal.
SUBROUTINE GETKN(TYPE, VALJE)
GETS NEXT LEXICAL ELEMENT IN USER'S COMMAND INPUT

COMMON/ SYSTEM/SYSIN, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

COMMON/LCODES/LEND, WORD, NJM, LP, RP
INTEGER WORD, RP

INTEGER TYPE, VALUE, CHAR, CARD, COL, ALPHNM, DIGIT, CHARBL, CHARLP,
1CHARRP, CHARAM, CHARST

DIMENSION VALUE(8), CARD(81), ALPHNM(39), LETTER(26), DIGIT(12)

EQUIVALENCE (LETTER, ALPHNM), (DIGIT, ALPHNM(27)),
1(IIXVAL, IYVAL)

DATA ALPHNM(1)/1HA/, ALPHNM(2)/1HB/, ALPHNM(3)/1HC/,
1ALPHNM(4)/1HD/, ALPHNM(5)/1HE/, ALPHNM(6)/1HF/, ALPHNM(7)/1HG/,
2ALPHNM(8)/1HH/, ALPHNM(9)/1HI/, ALPHNM(10)/1HJ/, ALPHNM(11)/1HK/,
3ALPHNM(12)/1HL/, ALPHNM(13)/1HM/, ALPHNM(14)/1HN/, ALPHNM(15)/1HO/,
4ALPHNM(16)/1HP/, ALPHNM(17)/1HQ/, ALPHNM(18)/1HR/, ALPHNM(19)/1HS/,
5ALPHNM(20)/1HT/, ALPHNM(21)/1HU/, ALPHNM(22)/1HV/, ALPHNM(23)/1HW/,
6ALPHNM(24)/1HX/, ALPHNM(25)/1HY/, ALPHNM(26)/1HZ/

DATA DIGIT(1)/1H/, DIGIT(2)/1H/, DIGIT(3)/1H/, DIGIT(4)/1H/,
1DIGIT(5)/1H/, DIGIT(6)/1H/, DIGIT(7)/1H/, DIGIT(8)/1H/,
2DIGIT(9)/1H/, DIGIT(10)/1H/, DIGIT(11)/1H/, DIGIT(12)/1H/,
3CHARRP/1H/, CHARAM/1H/, CHARBL/1H/, CHARLP/1H/,
4CHARST/1H/

DATA CARD(81)/1H/

IF(TYPE .NE. LEND .AND. COL .LT. 81) GO TO 120
READ(SYSIN, 2) (CARD(I), I=1, 80)
FORMAT(80A1)

**** NEXT TWO LINES NOT NEEDED FOR INTERACTIVE MODE ****
WRITE (SYSOUT, 3) (CARD(I), I=1, 80)
FORMAT(1H , 80A1)

COL=0

5 COL=COL+1
IF(COL .LT. 81) GO TO 120
TYPE=LEND
RETURN

FIND NEXT NON-DELMITER

) CHAR=CARD(COL)
IF(CHAR .EQ. CHARLP) GO TO 115
I=LKP1(CHAR, ALPHNM, 38)
IF(I .NE. 0) GO TO 200
IF(CHAR .EQ. CHARAM) GO TO 100
IF(CHAR .EQ. CHARRP) GO TO 150
IF(CHAR .EQ. CHARBL) GO TO 160
IF(CHAR .NE. CHARST) GO TO 115
FOUND '*'

TYPE=WORD
DO 125 J=2,8
VALUE(J)=CHARBL
VALUE(I)=CHAR
COL=COL+1
RETURN

FOUND RIGHT PAREN

) TYPE=RP
COL=COL+1
RETURN

FOUND LEFT PAREN

) TYPE=LP
COL=COL+1
RETURN

FOUND WORD

) IF(I .GT. 26) GO TO 300
TYPE=WORD
DO 210 J=2,8
VALUE(J)=CHARBL
J=0
) J=J+1
IF (J .GT. 8) GO TO 230
VALUE(J)=CHAR
) COL=COL+1
CHAR=CARD(COL)
IF(LKP1(CHAR,ALPHNM,38) .NE. 0) GO TO 220
RETURN

FOUND NUM OR '*-

) TYPE=NUM
I=I-26
IVAL=0
ISIGN=1
IF(I .NE. 12) GO TO 310
ISIGN=-1
IF(CARD(COL+1) .NE. CHARST) GO TO 320
COL=COL+2
TYPE=WORD
DO 305 J=3,8
) VALUE(J)=CHARBL
VALUE(I)=CHAR
VALUE(2)=CHARST
RETURN

GET INTEGER VALUE

) IF(I .EQ. 11) GO TO 350
IVAL=IVAL*10+I-1
) COL=COL+1
CHAR=CARD(COL)
I=LKPI(CHAR,DIGIT,11)
IF (I .NE. 0) GO TO 310
IVAL=IVAL*ISIGN
VALUE(1)=IVAL
XVAL=IVAL
VALUE(2)=IXVAL
RETURN

GET REAL VALUE (WITH FRACTION, IF PRESENT)

) XVAL=IVAL
POWER=1.*
) COL=COL+1
CHAR=CARD(COL)
I=LKPI(CHAR,DIGIT,10)
IF(I .EQ. 0) GO TO 370
POWER=POWER*10.*
XVAL=XVAL+(I-1)/POWER
GO TO 360
) XVAL=XVAL*ISIGN
VALUE(2)=IXVAL
VALUE(1)=IVAL*ISIGN
RETURN
END
Function LKPI

Function LKPI determines the relative position of a one-word argument in a list. Parameter LIST is the list to be searched for IARG. N is the number of entries in LIST. The function value returned is the relative position of IARG in LIST or zero if IARG is not found in LIST.

FUNCTION LKPI(IARG, LIST, N)
C DETERMINES WHETHER IARG IS IN LIST, AND, IF SJ, WHERE
C
DIMENSION LIST(N)
LKPI=0
IF(N .EQ. 0) RETURN
DO 10 I=1,N
IF(IARG .EQ. LIST(I)) GOTO 20
10 CONTINUE
RETURN
20 LKPI=I
RETURN
END
Function LKP8

Function LKP8 is called to determine the relative position of an eight-character name in a list of names. ARG is the name to be found, LIST is the list to be searched, and N is the number of entries in LIST.

The value returned is the position of ARG in LIST, or zero if ARG is not in LIST. Note that this function is frequently invoked with both ARG and LIST as parts of one-dimensional arrays.

```
FUNCTION LKP8(ARG, LIST, N)
C DETERMINES WHETHER ARG IS IN LIST, AND, IF SO, WHERE
C
REAL LIST
DIMENSION ARG(8), LIST(8, N)
L=0
LKP8=0
NT=N
10 IF(NT .EQ. 0) RETURN
   L=L+1
   DO 20 I=1,8
      IF(ARG(I) .NE. LIST(I, L)) GO TO 30
   CONTINUE
20   LKP8=L
   RETURN
30   NT=NT-1
   GO TO 10
END
```
Subroutine MOVE

Subroutine MOVE is called to move N words from array S to array T.
S and T frequently represent parts of larger arrays.

```
SUBROUTINE MOVE(S,T,N)
  C
  C MOVES N WORDS FROM ARRAY S TO ARRAY T
  C
  DIMENSION S(N),T(N)
  IF(N .LE. 0) RETURN
  DO 10 I=1,N
    10 T(I)=S(I)
  RETURN
END
```
Subroutine GETTOP

This subroutine operates like GETBOT, except that allocated storage is obtained from the top of array DATA and is not initialized when allocated.

SUBROUTINE GETTOP(N,L)
C ALLOCATES STORAGE AT TOP OF DATA ARRAY
C
COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWDRS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICDAE(11000)
EQUIVALENCE(ICDAT,CDAT)
C
COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT
C
L=TOP-N
TOP=L
IF(TOP .GT. BOT) RETURN
WRITE(SYSOUT,1)
FORMAT(1,*** INSUFFICIENT TEMPORARY STORAGE - EXECUTION*,
1 TERMINATED*)
STOP
END
Subroutine READ

Subroutine READ implements the READ command. Its function is to read selected data from the data base and to make these data available to subsequent commands. See Chapter IV for a description of the organization of the data after they have been read.

READ calls GTDSPC to scan the command qualifier and MRGORD to set the default output order for subsequent DISP commands. Storage is then obtained for the various "work" flags (see Table 2, Chapter IV) and all "read" and "work" flags are initialized. If an overlay tour has been specified in the command qualifier, a check is performed to insure that the overlaid tours have also been specified.

When reading data, precincts are selected on the basis of whether or not their precinct or division name appears in the qualifier. If no precinct or division names are specified, then all precincts are selected. Instead of storing a division name for each precinct read, a division number that refers to an entry in table LDIVNM (see Chapter IV) is used.

Within selected precincts, days are selected on the basis of the values of entries in table LDYRFL (see Chapter IV). For each day, hourly call rates and service times are computed from the corresponding parameters and hourly factors; service times are converted from minutes to hours.

Within days, tours are selected on the basis of the values of entries in table LTRRFL. A "tour type" is determined for each shift on the basis of its relationship to overlays (see Table 2). To facilitate indexing through the data, we have required that the same number of tours be stored for each day read, regardless of whether the day has an overlay tour or not. Therefore, the type of a tour is set to "ignore" when it occupies a position that would be held by an overlay tour, but the data base indicates that there is no overlay tour for the day. The meanings of other type codes should be apparent from Table 2.

Blocks are selected by entries in table LBLRFL. The constraint indicator for each block is set to -1 when it is read to indicate that
it has not been in the scope of a prescriptive command (MEET, ADD, or ALOC).

When the data for all tours and blocks of a day have been read, subroutine DERIVE is called to determine the numbers of actual and effective cars on duty in each block and to insure that enough cars are available in each hour of the day to handle the call-for-service workload.
SUBROUTINE READ

UBROUTINE TO READ SELECTED DATA FROM DATA BASE

COMMON/STATS/T(4,8),S(4,8),PORDER(3),ORDER(3),CIND(8)
INTEGER PORDER,ORDER

COMMON/SYSTEM/SYSIN,SYSCUT,IFILE,LIT
INTEGER SYSIN,SYSCUT

COMMON/OFFSET/NMPOFF,DVPOFF,APPOFF,SPPOFF,B1POFF,B2POFF,DYPHOFF,
1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CDOFF,STDOFF,TRDOFF,NWDOY,
2QDOFF,QXTOFF,CRTOFF,QTOFF,QMTOFF,CTOFF,QTOFF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,MTTOFF,LTTOFF,NPRIO,NWDT1,BLDOFF,QOBOFF,QNDFF,
4EBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBFFFF,CTBOFF,NWDOL
INTEGER DVPOFF,APPOFF,SPPOFF,B1POFF,B2POFF,DYPHOFF,CPDOFF,
1SPDOFF,OVDOFF,CDOFF,STDOFF,TRDOFF,QMTOFF,CTOFF,QTOFF,
2QDOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
3EBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBFFFF,CTBOFF,QOBOFF

COMMON/KEYWDS/NKEYW,DNTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER TYPOFF,WDTYPE
DIMENSION PCLSNNM(8),DCLSNN(8),TOUNRN(8)
EQUIVALENCE (PCLSNNM,KEYWD(1,4)),(DCLSNN,KEYWD(1,3)),
1(TOUNRN,KEYWD(1,2))

COMMON/PNTRS/IOVRLY,IOVTR(2),
1NPCTDT,NPCTR,F,LPCD,NNMLST(4),NNAMES(4),NDAVTR,LDAYNM,
2RORF,LDAVRI,LDYWFL,NTROT,LTTRTB(2),LRTST,LRSTN,LTRTRL,LTRN,
3NTRR,LTRWFL,NLBLO,LBLKTB(2),BLBRFL,NBLRD,LBLWFL,NDJVTR,LNDVRD,
4LDIVNM,LDIVFL

COMMON/STORE/TOPT,TOT,RDBOT,MAXBOF,NWDRS,CDAT(11000)
INTEGER TOP,TOT,RDBOT
DIMENSION IDCAT(11000)
EQUIVALENCE (ICDAT,CDAT)

COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

INTEGER TYPE,VAL,ORDER,OVRLY
DIMENSION VAL(2),TPCTNM(8),TDIVNM(8),TPCTDT(20),TDATA(24),
1ORDER(4)
EQUIVALENCE (TPCTNM,TPCTDT),(TDIVNM,TPCTDT(9)),
1(TPCTDT,TDATA),(OVRLY,IOVRLY)

INITIALIZE

NPCTRO=0
BOT=RDBOT
LGETB=BOT
LGETT=TOP

581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
NDIVRQ=0
NDIVRD=0
NPCTR=0
NDAYRD=0
NTRRD=0
NBLRD=0

INTERPRET QUALIFIERS

TYPE=CMD
CALL SCAN(TYPE,VAL)
CALL GTDSC(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 10
TOP=LET
RETURN

SET DEFAULT OUTPUT ORDER FOR DISP

CALL MRGORD(ORDER,ORDER,ORDER)

GET STORAGE FOR WORK FLAGS.
INITIALIZE FLAGS
DATA FOR DAYS

IT=1
L=LMLST(IT)
N=NNAMES(IT)
IF(N .NE. 0) GO TO 30
CALL GETBOT(NDAYDT,LDYWF)
DO 20 I=1,NDAYDT
ICDAT(LDYWF+I-1)=I
ICDAT(LDYRF+I-1)=I
NDAYRD=NDAYDT
GO TO 100
CALL GETBOT(N,LDYWF)
DO 40 I=1,NDAYDT
ICDAT(LDYRF+I-1)=0
DO 50 I=1,N
J=LKB8(ICDAT(L),ICDAT(LDAYNM),NDAYDT)
IF(J .EQ. 0) GO TO 900
ICDAT(LDYRF+J-1)=I
L=L+8
NDAYRD=N
IDAY=0
DO 60 I=1,NDAYDT
IF(ICDAT(LDYRF+I-1) .EQ. 0) GO TO 60
IDAY=IDAY+1
ICDAT(LDYRF+I-1)=IDAY
ICDAT(LDYWF+IDAY-1)=I
CONTINUE

DATA FOR TOURS

IT=2
N=NNAMES(IT)
IF(N .NE. 0) GO TO 120
CALL GETBOT(NRD,LTRWF)
DO 110 I=1,NRD

638 639 640 641 642 643 644
645 646 647 648 649 650 651
652 653 654 655 656 657 658
659 660 661 662 663 664 665
666 667 668 669 670 671 672
673 674 675 676 677 678 679
680 681 682 683 684 685 686
687 688 689 690 691 692 693
694 695
ICDAT(LTRWFL+I-1)=I
ICDAT(LTRRFL+I-1)=I
NTRRD=NTRDT
GO TO 200
L=LNMLST(IT)
DO 130 I=1,NTRDT
ICDAT(LTRRFL+I-1)=0
CALL GETBOT(N,LTRWFL)
DO 140 I=1,N
J=LKP8(CDAT(L),CDAT(LTRNM),NTRDT)
IF(J .EQ. 0) GO TO 900
ICDAT(LTRRFL+J-1)=1
L=L+8
NTRRD=N
ITOUR=0
DO 150 ITYPE=1,NTRDT
IF(ICDAT(LTRRFL+ITYPE-1) .EQ. 0) GO TO 150
ITOUR=ITOUR+1
ICDAT(LTRRFL+ITYPE-1)=ITOUR
ICDAT(LTRWFL+ITOUR-1)=ITYPE
CONTINUE

DATA FOR DIVISIONS

IT=3
N=NNAMES(IT)
NDIVREQ=N
IF(N .EQ. 0) GO TO 300
IF(N .GT. NDIVDT) GO TO 910
L=LNMLST(IT)
CALL MOVE(CDAT(L),CDAT(LDIVNM),8*N)

IT=4
NPCTRQ=NNAMES(IT)
IF(NPCTRQ .GT. NPCTDT) GO TO 910
LPCTNM=LNMLST(IT)

DO 305 I=1,NBLDT
ICDAT(LBLRFL+I-1)=0
N=NTRDT
IF(OVRLY .NE. 0) N=N-1
DO 315 I=1,N
IF(ICDAT(LTRRFL+I-1) .EQ. 0) GO TO 315
D3 310 J=1,2
K=ICDAT(LTRTB(J)+I-1)
IF(K .EQ. 0) GO TO 315
NBLRD=NBLRD+1
ICDAT(LBLRFL+K-1)=NBLRD
CONTINUE
CONTINUE

CHECK OVERLAY TOURS

IF(OVRLY .EQ. 0 .OR. ICDAT(LTRRFL+NTRDT-1) .EQ. 0) GO TO 340
IBLK1=ICDAT(LTRTB(1)+NTRDT-1)
IBLK2=ICDAT(LTRTB(2)+NTRDT-1)
I=LKP1(IBLK1,ICDAT(LTRTB(2)),NTRDT)
IF(I .NE. 0) GO TO 325
N1 = 1
! WRITE(SYSPUT,9) N1,TOURNM
STOP
! IOVTR(1)=ICDAT(LTRRFL+I-1)
IF(IOVTR(1) .EQ. 0) GO TO 320
I=LP1(IBLK2,ICDAT(LTRTB(1)),NTRDT)
IF(I .NE. 0) GO TO 335
N2 = 2
! WRITE(SYSPUT,9) N2,TOURNM
STOP
! IOVTR(2)=ICDAT(LTRRFL+I-1)
IF(IOVTR(2) .EQ. 0) GO TO 330
! NWDDY=TRDOFF+NTRRD*NWDTN+BLLRD*NWDBL
NWDPCT=DYP0FF+NDAYRD*NWDDY
NDIVRP=NDIVRQ
BLDOFF/TRDOFF+NTRRD*NWDTN
REWIND IFILE
CALL SKIP(IFILE,(1+(NDAYKT-1)/10+1+NTRDT))

READ PRECINCT HEADER RECORD

DO 450 IPCT=1,NPCTDT
READ(IFILE,1) TPCTDT
IF(NPCTRQ=NDIVRP .NE. 0) GO TO 350
IDIV=LP8(TDIVNM,CDAT(LDIVNM),NDIVRP)
IF(IDIV .NE. 0) GO TO 370
NDIVRDP=NDIVRP+1
IF(NDIVRP .LE. NDIVDT) GO TO 347
WRITE(SYSPUT,8) OCLS
STOP
CALL MOVE(TDIVNM,CDAT(LDIVNM+(NDIVRP-1)*8),8)
IDIV=NDIVRP
GO TO 370

IF(NDIVRP .EQ. 0) GO TO 355
IDIV=LP8(TDIVNM,CDAT(LDIVNM),NDIVRP)
IF(IDIV .NE. 0) GO TO 370
IF(NPCTRQ .EQ. 0) GO TO 360
J=LP8(TPCPNM,CDAT(LPCTNM),NPCTRQ)
IF(J .NE. 0) GO TO 345
CALL SKIP(IFILE,NDAYKT*(4+NTRDT))
GO TO 450
NPCTRDP=NPCTRQ+1
CALL GETB0T(DYP0FF,LPCT)
IF(NPCTRQ .EQ. 1) LPCTDT=LPCT
CALL MOVE(TPCPNM,CDAT(LPCT+NP0FF),8)
ICDAT(LPCT+DYP0FF)=IDIV
CALL MOVE(TPCPTAT17,CDAT(LPCT+ARPOFF),4)

READ DAY DETAIL RECORDS FOR THIS PRECINCT

DO 440 IDAY=1,NDAYKT
IF(IDCALT(DYRFL+IDAY-1) .NE. 0) GO TO 375
CALL SKIP(IFILE,(4+NTRDT))
GO TO 440
CALL GETB0T(TRDOFF,LDAY)
LCR=LDAY*CRDOFF-1
READ(IFILE,2) CDAT(LDAY+CPDOFF), CDAT(LDAY+SPDOFF),
C ICDAT(LDAY+OVDOFF), (ICDAT(LCR+I), I=1,48)
CPARM=CDAT(LDAY+CPDOFF)
SPARM=CDAT(LDAY+SPDOFF)

CALCULATE CALL RATES AND SERVICE TIMES

DO 380 I=1,24
  I=I-1
  CDAT(LDAY+CRDOFF+I)=CDAT(LDAY+CRDOFF+I)*SPARM
  CDAT(LDAY+STDOFF+I)=CDAT(LDAY+STDOFF+I)*SPARM/60.

READ SHIFT DETAIL RECORDS
FOR THIS DAY AND PRECINCT

DO 400 ITOUR=1,NTRDT
  IF(ICDAT(LTRRFL+ITOUR-1) .NE. 0) GO TO 390
  CALL SKIP(IFILE,1)
  GO TO 400
  CALL GETBOT(NWDR,T,LTUR)
  READ(IFILE,3) (CDAT(LTUR+ACTOFF+I-1), I=1,5)
  CDAT(LTUR+XTOFF)=-1.
  ICDAT(LTUR+TYTOFF)=2
  IF(DVRL .EQ. 0 .OR. ITOUR .LT. NTRDT) GO TO 400
  IF(ICDAT(LDAY+OVDOFF) .NE. 0) GO TO 395
  ICDAT(LTUR+TYTOFF)=1
  GO TO 410
  ICDAT(LTUR+TYTOFF)=5
  ICDAT(LDAY+TRDOFF+(IODVTR(1)-1)*NWDR+TYTOFF)=3
  ICDAT(LDAY+TRDOFF+(IODVTR(2)-1)*NWDR+TYTOFF)=4
  GO TO 410
  CONTINUE

READ BLOCK DETAIL RECORD
FOR THIS DAY AND PRECINCT

READ(IFILE,4) TDATA
DO 420 I=1,NBLDT
  IF(ICDAT(LBLRFL+I-1) .EQ. 0) GO TO 420
  CALL GETBOT(NWDBL,LBLOCK)
  CDAT(LBLOCK+CBDOFF)=TDATA(I)
  ICDAT(LBLOCK+CTBOFF)=-1
  CONTINUE

CHECK THAT MINIMUM ALLOCATION IS PRESENT
AND CALCULATE AVERAGES

CALL DERIVE(LPCT,LDAY)
CONTINUE
CONTINUE
IF(NPCTR .GT. 0) GO TO 460
WRITE(SYSOUT,7) PCLSNM
TOP=LET
BOT=LETB
RETURN
CALL GET90T(NDIVRD,LDIVFL)
TOP=LET
RETURN
WRITE(SYSOUT,5) (KEYWD(I,IT), I=1,8), (CDAT(L+I-1), I=1,8)
GO TO 920
WRITE(SYSOUT,6) (KEYWD(I,IT),I=1,8)
TOP=LGETT
BOT=LGETB
RETURN

FORMAT FOR PRECINCT HEADER RECORD
FORMAT(8A1,1X,8A1,1X,4(1X,F5.0))
FORMAT FOR DAY DETAIL RECORDS
FORMAT(2(F5.0,1X),11/24F3.2/24F3.2)
FORMAT FOR SHIFT DETAIL RECORD
FORMAT(13(F5.0,1X))
FORMAT FOR BLOCK DETAIL RECORD
FORMAT(24F3.1)
FORMAT(/' ***',2(1X,8A1),' NOT IN DATA - REENTER')
FORMAT(/' *** TOO MANY ',8A1,' S SPECIFIED - REENTER')
FORMAT(/' *** NO ',8A1,' DATA SELECTED - REENTER.')
FORMAT(/' *** BLOCK ',11,' FOR OVERLAY ',8A1,' NOT FOUND',
      ' - EXECUTION TERMINATED')
FORMAT(/' *** DATA BASE ERROR: MORE UNIQUE ',8A1,' NAMES THAN',
      ' DECLARED - EXECUTION TERMINATED')
END
Subroutine GTDSPC

Subroutine GTDSPC (get data specification) is called to scan command qualifiers. It obtains up to four lists of names which are the user's specifications for days, tours, divisions, and precincts. The lists are stored in array CDAT. List pointers are stored in array LNMLST and list lengths are stored in array NNAMES (see Chapter IV).

GTDSPC parameters TYPE and VAL are passed to subroutine SCAN, which is called to obtain syntactic elements from the input stream. Thus, GTDSPC's calling program can determine what syntactic element followed the qualifier in the input stream. At entry, GTDSPC assumes that TYPE and VAL have been set by a previous call to SCAN so that they describe the first element of the qualifier, or the next input element if the qualifier is null.

GTDSPC also returns a three-element array (ORDER) that specifies the order of the phrase types in the qualifier (which in turn determines the DISP command default output order). The elements of ORDER correspond to phrases in the qualifier, e.g., ORDER(1) refers to the first phrase, ORDER(2) to the second phrase, ORDER(3) to the third phrase. The values of the elements of ORDER indicate the type of phrase in each position as follows: 1 = DAY phrase; 2 = TOUR phrase; 3 = DIVISION or PRECINCT phrase (the numbers are derived from the order of the keywords in table KEYWD; DIVISION and PRECINCT are considered equivalent in this context, and the second one entered is ignored).
SUBROUTINE GTDSPC(TYPE, VAL, ORDER)

GETS DATA SPECIFICATION BY SCANNING QUALIFIERS

COMMON/SYSTEM/SYSIN, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

COMMON/KEYWDS/NKWOD, NTYPES, TYPEOFF(4), KEYWD(8, 30), WDTYPE(30)
INTEGER TYPEOFF, WDTYPE
DIMENSION PCLSN(8), DCLSNM(8), TOURNM(8)
EQUIVALENCE (PCLSN, KEYWD(1, 4)), (DCLSNM, KEYWD(1, 3)),
( TOURNM, KEYWD(1, 2))

COMMON/PNTRS/IOVRLY, IOVTR(2),
INPCTDT, NPCTRD, LPCTDT, LNMLST(4), NNAME(4), NDAYDT, LDAYNM,
2LDYRF, NDAYRD, LDYWF, NTRDT, LTRTB(2), LTRST, LTRND, LTRRF, LTRNM,
3NTRD, LTRWF, NBLDT, LBLKT(2), LBLRF, NBLRD, LBLWF, NDIVDT, NDIVRD,
4LDIVNM, LDIVFL

COMMON/SCODES/SEND, CMD, NUMLST, NAMLST, FSPEC, DSPEC, DUM, ERR
INTEGER SEND, CMD, FSPEC, DSPEC, DUM, ERR

INTEGER TYPE, VAL, ORDER
DIMENSION VAL(2), ORDER(3)
LGETT=TOP
ORDER=0
DO 5 I=1, 3
ORDER(I)=0
NNAMES(I)=0
NNAMES(4)=0
GO TO 12

GET PHRASE TYPE

CALL SCAN(TYPE, VAL)
IF(TYPE .EQ. DSPEC) GO TO 15
IF(TYPE .EQ. ERR .OR. TYPE .EQ. FSPEC .OR. TYPE .EQ. SEND)
1 RETURN
WRITE(SYSOUT, 2)
FORMAT(15 *** INVALID QUALIFIER - REENTER*)
TYPE=ERR
RETURN

KEYVAL=VAL(1)
IT=KEYVAL-TYPEOFF(DSPEC)

GET NAME LIST

CALL SCAN(TYPE, VAL)
IF(TYPE .EQ. NAMLST) GO TO 20
WRITE(SYSOUT, 1)(KEYWD(I, KEYVAL), I=1, 8)
FORMAT(15 *** INVALID ', 8AL', SPECIFICATION - REENTER*)
TYPE=ERR
TOP=LGETT
RETURN

TERMINE OUTPUT ORDER SPECIFIED BY THIS COMMAND
LNMLST(IT)=VAL(2)
NAMES(IT)=VAL(1)
IORDER=IORDER+1
IF(IORDER .GT. 3) GO TO 10
IF(IT .EQ. 4) IT=3
IF(LKP1(IT,ORDER,3) .NE. 0) GO TO 10
ORDER(IORDER)=IT
GO TO 10
END
Subroutine MRGORD

Subroutine MRGORD (merge order) is called to set a new default output order from the qualifier of a READ or DISP command. Its arguments are arrays which fit the description of the ORDER parameter of subroutine GTDSPC. NEWORD represents the ordering of qualifier phrases in the last qualifier scanned. OLDORD represents the previously existing ordering of output phrases. OUTORD represents an ordering of output phrases resulting from merging the "old" ordering with the "new" ordering.

Subroutine MOVE is called to move the contents of OLDORD to a temporary storage (TMPORD) where elements can be "erased" without affecting the original values in OLDORD. Elements of NEWORD are moved to OUTORD in their current order and the phrase types moved are erased from TMPORD. Any elements of OUTORD left unfilled by this process are filled by moving elements from TMPORD in the order in which they occur. Thus, a new DISP command output order is established.

```fortran
SUBROUTINE MRGORD(NEWORD, OLDORD, OUTORD)
C SETS OUTPUT ORDER FOR DISP COMMAND. MERGES NEW INFORMATION INTO OLD TO ESTABLISH OUTPUT ORDER
C
INTEGER OLDORD, OUTORD, TMPORD
DIMENSION NEWORD(3), OLDORD(3), OUTORD(3), TMPORD(3)

CALL MOVE(OLDORD, TMPORD, 3)
DO 30 IORD=1,3
IF(NEWORD(IORD) .EQ. 0) GO TO 10
OUTORD(IORD)=NEWORD(IORD)
I=LKPI(NEWORD(IORD), TMPORD, 3)
IF(I .NE. 0) TMPORD(I)=0
GO TO 30
10
DO 20 I=1,3
IF(TMPORD(I) .EQ. 0) GO TO 20
OUTORD(IORD)=TMPORD(I)
TMPORD(I)=0
GO TO 30
20 CONTINUE
RETURN
30 CONTINUE
RETURN
END
```
Subroutine DERIVE

Subroutine DERIVE is called by READ and SET after reading or modifying the data for a day for a precinct. Its two parameters LPCT and LDAY are pointers to the data for the precinct and the day, respectively. DERIVE's primary function is to determine, for each block of the day: number of actual cars on duty; average cfs workload; maximum cfs workload in any hour; total calls for service; and number of effective cars on duty. For each tour of the day, DERIVE determines the fraction of calls in the lowest priority class, and the total number of calls during the tour.

Subroutine SBLACT (set block actual cars) is called to determine the number of actual cars on duty in each block of a day from the number of cars assigned to each tour of the day. The number of effective cars on duty in a block is computed using the formula given in Chapter III.

DERIVE also checks to determine whether or not each block has enough effective cars to handle the call-for-service workload in its busiest hour. If a block lacks sufficient effective cars, then the algorithm described in Chapter III is used to determine where to increase the assignment of cars to tours of a day so that each block will have enough effective cars. After this algorithm has been applied, subroutine SBLEF (set block effective cars) is called to redetermine the number of effective cars in each block of the day.
SUBROUTINE DERIVE(LPCT, LDAY)

CALCULATES, FOR EACH BLOCK IN DAY AND PRECINCT,
AVERAGES OF INPUT DATA

COMMON KEYWD$NKYWD$, NTYPES, TYPOFF(4), KEYWD(8, 30), WDTYPE(30)
INTEGER TYPOFF, WDTYPE
DIMENSION PCLSNM(8), DCLSNM(8), TOURNM(8)
EQUIVALENCE (PCLSNM, KEYWD(1, 4)), (DCLSNM, KEYWD(1, 3)),
1(TOURN, KEYWD(1, 2))

COMMON/STOR$SYSIN$, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

COMMON/STORE$TOP$, BDT, RBDT, MAXBDT, NWDRS, CDAT(11000)
INTEGER TOP, BDT, RBDT
DIMENSION ICODAT(11000)
EQUIVALENCE (ICODAT, CDAT)

COMMON/NPOFF, DMPOFF, ARPFF, SMPOFF, B1POFF, B2POFF, DYPOFF,
WPOFF, CPPOFF, SPPOFF, OVPOFF, CRPOFF, STPOFF, TRPOFF, NWPOFF,
2QTOFF, XTTOFF, CTTOFF, QTTOFF, QNTOFF, CTITOFF, YITTOFF, ACTOFF, RVTOFF,
3PITTOFF, HFTOFF, MFTOFF, LTTOFF, NPI, NWTR, BLDOFF, QBOFF, QNB0FF,
4FBOFF, ACBOFF, ABNOFF, CRBOFF, RMBOFF, OCBOFF, CBBOFF, NWBL

INTEGER DPOFF, ARPFF, SMPOFF, B1POFF, B2POFF, DYPOFF, CPPOFF,
1WPOFF, OVPOFF, CRPOFF, STPOFF, TRPOFF, QTTOFF, QNTOFF, CTITOFF, YITTOFF,
2QTOFF, XTTOFF, CTTOFF, QTTOFF, ACTOFF, RVTOFF, HFTOFF, BLDOFF,
3PITTOFF, ABNOFF, CRBOFF, RMBOFF, OCBOFF, CBBOFF, CTBOFF, QBOFF, QNB0FF

COMMON/PRNTR$IOVRLY$, IOVTR(2),
1NPCTD, NPCTR, LPCTD, LNMLST(4), NNAMEST(4), NDAYT, LDAYNM,
2DT, NDAYRD, LNWFL, NRTDR, LTRTB(2), LRTST, LTRNDR, LTRNTLM,
3TNRD, LTRWFL, NLBDT, BLKTB(2), LBLRFL, N8RD, LBLWFL, NDIVDT, NDIVRD,
4LIVNM, LIVFL

REAL LFR
DIMENSION IBERR(24), ACB(2), ACTR(2)

IDERR=0
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)

FIND NUMBER OF CARS ON DUTY IN EACH BLOCK

CALL SBLACT(LPCT, LDAY)
DO 20 IBLDT=1, NBLDT
IBLKL=ICODAT(LBLRFL+IBLDT-1)
IF(IVBLKL=0) GO TO 20
LBDKLDAY+BDLOFF(IKL-1)*NWBL
IBERR(IBLK)=0
ISTART=ICODAT(LBLKTB(1)+IBLDT-1)
IEND=ICODAT(LBLKTB(2)+IBLDT-1)
BLKLN=IEND-START+1

CALCULATE AVERAGE AND MAXIMUM CALL RATE IN BLOCK
RMAX=0.
CRATE=0.
AWL=0.
LB=LDAY-1
DO 10 I=ISTART,IEND
IB=I+LB
CR=CDAT(IB+CRDOFF)
CRATE=CR-ATE+CR
R=CR*CDAT(IB+STDFF)
IF(R.GT.RMAX) RMAX=R
AWL=AWL+R
AWL=AWL/BLKL
CDAT(LBLK+CRBOFF)=CRATE
CDAT(LBLK+AWBOFF)=AWL
ACT=CDAT(LBLK+ACBOFF)

CALCULATE EFFECTIVE CAS AND CHECK WHETHER
MINIMUM ALLOCATION IS ACHIEVED

EF=ACT*(1.-(B1*AWL/ACT)+B2))
NEF=EF
IF(NEF.GT.RMAX) GO TO 15
IBERR(IBLK)=1
ACT=CEIL((EF+B1*AWL)/(1.-B2))
EF=ACT*(1.-(B1*AWL/ACT)+B2))
NEF=EF
IF(NEF.GT.RMAX) GO TO 13
ACT=ACT+1.
GO TO 12
CDAT(LBLK+ACBOFF)=ACT
CDAT(LBLK+EFBOFF)=EF
CDAT(LBLK+RMBOFF)=RMAX
CONTINUE

DO 100 ITYPE=1,NTRDT
ITOUR=ICDAT(LTRRFL+ITYPE-1)
IF(ITOUR.LT.1) GO TO 100
ITERR=0
LTOUR=LDAY+TRDOFF+(ITOUR-1)*NWDTTR
IF(ICDAT(LTOUR+TYTOFF).EQ.1) GO TO 100

CALL RATE IN TOURS

IF(NP0.I.LT.2) GO TO 40
LFR=1.
N=NPRIO-1
DO 30 I=1,N
LFR=LFR*CDAT(LTOUR+HTOFF+I-1)
CDAT(LTOUR+HTOFF+N)=LFR
CRATE=0.
DO 50 IBLK=1,2
ACB(IBLK)=0.
IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
IF(IELD.LT.1) GO TO 50
IBLR=ICDAT(LBLRFL+IBLD-1)
LBLK=LDAY+BLDOFF+(IBLR-1)*NWDBL
ACB(IBLK)=CDAT(LBLK+ACBOFF)

1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096
CRATE=CRATE+CDAT(LBLK+CRBFF)
IF(1BERR(IABLX) .EQ. 0) GO TO 50
ITERR=ITERR+1BLK
CONTINUE

CDAT(LTOUR+CROFF)=CRATE
CALCULATE NEW NUMBER OF ACTUAL CARS IN TOURS,
IF THERE HAS BEEN A CHANGE IN THE BLOCKS

ID=ICDAT(LTOUR+TYOFF)-1
GO TO (60, 65, 70, 75), ID
IF(1TER' .EQ. 0) GO TO 90
ACT=AMAXI(ACB(I),ACB(2))
GO TO 85
ACTR(I)=ACB(I)
IF(1TER' .EQ. 2 OR 1TER' .EQ. 0) GO TO 90
ACT=ACB(I)
GO TO 85
ACTR(2)=ACB(2)
IF(1TER' .EQ. 1 OR 1TER' .EQ. 0) GO TO 90
ACT=ACB(2)
GO TO 85
IF(1TER' .EQ. 0) GO TO 90
ACT=CDAT(LTOUR+ACTOFF)
TACT=ACT
DO 80 I=1,2
ACT=AMAXI(ACT,ACB(I)-ACTR(I))
IF(TACT .GE. ACT) GO TO 90
LPNM=(LPCT+NMPOFF-1)
LTMN=LTRNM+((ITYPE-1)*8-1
IDAY=(LDAY-DPOFF-LPC1)/NWDY
IDAY=ICDAT(LDYWFL+IDAY)
LDNM=LDAYNM+(IDAY-1)*8-1
WRITE(SYSOUT,2) ACT,PCLSNM,(ICDAT(LPNM+I),I=1,8),
LTOURNM,(ICDAT(LTMN+I),I=1,8),(ICDAT(LDNM+I),I=1,8)
FORMAT(2(1X,8A1)," ON DAY ",8A1)
CDAT(LTOUR+ACTOFF)=ACT
IDERR=1
CONTINUE
CONTINUE

IF(IDERR .EQ. 0) RETURN

REDETERMINE EFFECTIVE CARS IF CHANGE IN ACTUAL CARS

CALL SBLACT(LPCT,LDAY)
CALL SBLEF(LPCT,LDAY)
RETURN
END
Subroutine SBLACT

Subroutine SBLACT (set block actual cars) is called to determine the number of actual cars on duty in each block of a day in a precinct, based on the number of cars assigned to each tour of the day. Parameters LPCT and LDAY are pointers to the data for the precinct and day for which the block allocations are to be determined. The algorithm used to determine the number of cars on duty in each block of a day from the number of cars on duty in each tour of the day is given in Chapter III.
SUBROUTINE SBLACT(LPCT, LDAY)

CALCULATES NUMBER OF ACTUAL CARS IN BLOCKS,
BASED ON NUMBER OF CARS IN TOURS.

COMMON/STORE/TCP, BOT, RDBOT, MAXBOT, NW3ROS, CDAT(1000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICADAT(1000)
EQUIVALENCE (ICADAT, CDAT)

COMMON/PNTRS/IOVRLY, IOVTR(2),
 1NPCTDT, NPCTRD, LPCTDT, LNMLST(4), NNAMES(4), NDAYDT, LDAYNM,
 2LDYRFL, NDAYRD, LDYWFL, NTRDT, LTRT8(2), LTRST, LTREND, LTRRFL, LTRNM,
 3NTRRD, LTRWFL, NBLDT, LBLKTB8(2), LBLRFL, NBLRD, LBLWFL, NDIYDT, NDIYRD,
 4LDIVNM, LDIVFL

COMMON/OFFSET/NMPOFF, DVPOFF, ARPOff, SMPOFF, B1POFF, B2POFF, DYPOFF,
 1NWDPT, CPDOFF, SPDOFF, OVDOFF, CRDOFF, STDOFF, TRDOFF, NWDOY,
 2QTOFF, QXTOFF, CRTOFF, QTOFF, QNTOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF,
 3PVTTOFF, HFTOFF, MFTOFF, LFTOFF, YPRIC, NWDTO, BLDOFF, QBOFF, QNBOFF,
 4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, QCBOFF, CTBOFF, NWDBL

INTEGER DVPOFF, ARPOff, SMPOFF, B1POFF, B2POFF, DYPOFF, CPDOFF,
 1SPDOFF, OVDOFF, CRDOFF, STDOFF, TRDOFF, QTOFF, QXTOFF, CRTOFF, QTOFF,
 2QTOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF, PVTTOFF, HFTOFF, MFTOFF, BLDOFF,
 3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, QCBOFF, CTBOFF, QBOFF, QNBOFF

DO 10 IBLK=1, NBLRD
LBLK=LDAY+BLDCFF*(IBLK-1)*NWDBL
10 CONTINUE

DO 30 ITYPE=1, NTRDT
ITOUR=ICADAT(LTRRFL+ITYPE-1)
IF((ITOUR .LT. 1)) GO TO 30
LTOUR=LDAY+TRDOFF+(ITOUR-1)*NWDTO
IF((ICADAT(LTOUR+TYTOFF)*EQ. 1)) GO TO 30
20 IB=1, 2
13LKD=ICADAT(LTRTB8(IB)+ITYPE-1)
IF((LBDKDT .EQ. 0)) GO TO 20
IBKRD=ICADAT(LBLRFL+IBLDKDT-1)
LBLK=LDAY+BLDCFF*(IBKRD-1)*NWDBL
CDAT(LBLK+ACBOFF)=CDAT(LBLK+ACBOFF)+CDAT(LTDUR+ACTOFF)

CONTINUE

RETURN
END
Subroutine SBLEF

Subroutine SBLEF (set block effective cars) determines the number of effective cars on duty in each block of a day. Parameters LPCT and LDAY are pointers to the data for the precinct and day for which the calculations are to be performed. Chapter II and Appendix B of the User's Manual give the formula used to compute effective cars from actual cars and average workload.

SUBROUTINE SBLEF(LPCT, LDAY)

CONVERTS ACTUAL CARS TO EFFECTIVE CARS IN EACH BLOCK

COMMON/PNTRS,IOVRLY,IOVTR(2),
1NPCTD, NPTD, LPCTDT, LNMLST(4), NNNAMES(4), NDAYDT, LDAYNM,
2LDYRFL,NDAYRD,LDWFL,NTRDT,LTTRB(2),LFRST,LTREND,LTTRFL,LTRNM,
3NTRRD,LTRWFL,NBLDT,LBLKT(2),LBLRFL,NBLRD,LBLWFL,NDIVOT,NDIVRD,
4LDIVNM, LDIVFL

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWJORDS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

COMMON/OFFSET/NMPOFF,DVPOFF,ARPFOFF, SPOFF, BPOFF, B2POFF, DYPOFF,
1NWDPCT, CPDOFF, SPDOFF, OVDOFF, CRDOFF, STDOFF, TDROFF, NWDDY,
2QTOFF, QTOFF, CTOFF, QTOFF, QTOFF, QTOFF, CTOFF, TTOFF, ATOFF, RTOFF,
3PVTOFF, HTOFF, FTOFF, ETOFF, NTOFF, NWHITE, BLOFF, QBOFF, QNBOFF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, NWDBL

INTEGER DVPCCF, ARPFOFF, SPOFF, BPOFF, B2POFF, DYPOFF, CPDOFF,
1SPDOFF, OVDOFF, CRDOFF, STDOFF, TDROFF, QTOFF, QTOFF, CTOFF, QTOFF,
2QTOFF, CTOFF, TTOFF, ATOFF, RTOFF, PVTOFF, HTOFF, FTOFF, BLOFF,
3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, QBOFF, QNBOFF

B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
DO 10 IBLK=1,NBLRD
LBLK=LDAY+BLOFF+(IBLK-1)*NWDBL
AWL=CDAT(LBLK+AWBOFF)
ACT=CDAT(LBLK+ACBOFF)
CDAT(LBLK+EFBOFF)=ACT*(1.-((B1*AWL/ACT)+B2))
RETURN
END
Subroutine LIST

Subroutine LIST implements the LIST command. It prints input data (and some derived values) for selected precincts, days, and tours.

Subroutine GTDSFC is called to scan the qualifier and SETWFL is called to define the subset of precincts, days, and tours for which data will be listed. Function NXPCT is called to set a pointer (LPCT) to the data for the next precinct selected. A pointer value of zero at entry to NXPCT requests the first precinct selected; a pointer value of zero returned from NXPCT means no more precincts have been selected. The name, area, street miles, and unavailability parameters are printed for each precinct selected.

After a precinct pointer has been obtained and the data for the precinct printed, function NXDAY is called to find the days for which data are to be listed. As with NXPCT, the value of the day data pointer (LDAY) at entry to NXDAY indicates whether the first day for a precinct is to be located, and the value of the day pointer returned from NXDAY is zero if there are no more days selected. For each selected day its name, call rate parameter, and service time parameter are printed; in addition, column headings are printed for the tour data which follow.

Function NXTOUR is used in the same manner as NXPCT and NXDAY to index through the tours of each day. For each tour selected, LIST computes average call rate and service time over all its hours and the average number of effective cars in its blocks. These are printed along with the tour name, actual cars assigned, response speed, patrol speed, and the fraction of calls in each priority class.
SUBROUTINE LIST

IMPLEMENTS THE LIST COMMAND

COMMON/STORE/STOP,BOT,ROBOT,MBOT,NCARDS,CDAT(11000)
INTEGER STOP,BOT,ROBOT
DIMENSION ICARD(11000)
EQUIVALENCE (ICARD,CDAT)

COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT

COMMON/KEYWDS/NKEYW,NTYPES,TYPOFF(4),KEYW(8,30),WDTYPE(30)
INTEGER TYPOFF,WDTYPE
DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYW(1,4),(DCLSNM,KEYW(1,3)),
1(TOURNM,KEYW(1,2))

COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
1NPDPC,TCPDFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDY,
2QDTOFF,QXTOFF,CRTOFF,QTOFF,QNTOFF,CTT(JFF),TYT0FF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIQ,NWDT,BLDOFF,QOBFF,QNBFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,QBBOFF,CTBOFF,NWDBL

INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QT0FF,QXTOFF,CRTOFF,QTOFF,
2QNTOFF,CTT(JFF),TYT0FF,ACTOFF,RVTOFF,PVTOFF,AFTOFF,BLDOFF,
3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,QBBOFF,CTBOFF,QOBFF,QNBFF

COMMON/PNTRS/I0VRLY,IOVTR(2),
1NPCTDT,NPCRD,LPCTDT,LMNLIST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRL,NDAYRD,LDYWFL,TRTDT,LRTRTB(2),LRTST,LTRTEND,LTRRFL,LTRNM,
3NTRTD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
4LDIVNM,LDIVFL

COMMON/SOCODES/SEND,CMN,NUMLST,NAMELIST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMN,FSPEC,DSPEC,DUM,ERR

DIMENSION VAL(2),ORDER(3)
INTEGER TYPE,VAL

LGETT=TOP
TYPE=CMD

INTERPRETS QUALIFIER OF LIST COMMAND

CALL SCAN(TYPE,VAL)
CALL GTDSPC(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 10
TOP=LGETT
RETURN
CALL SETWFL(IERR)
IF(IERR .EQ. 0) GO TO 15
TOP=LGETT
RETURN

FIND NEXT PRECINCT, WRITE HEADER INFORMATION
LPCT=0
LPCT=NXPCT(LPCT)
IF(LPCT NE 0) GO TO 30
TOP=LGETT
RETURN
WRITE(SYSOUT,1) PCLSNM, (ICDAT(LPCT+VMPOFF+I-1),I=1,8),
1 CDAT(LPCT+ARPOFF),CDAT(LPCT+SMPOFF),CDAT(LPCT+B2POFF),
2 CDAT(LPCT+B1POFF)

FIND NEXT DAY. LIST HEADER INFORMATION.

LDAY=0
LDAY=NXDAY(LPCT,LDAY)
IF(LDAY EQ 0) GO TO 20
LDAY=(LDAY-LPCT-DYPOFF)/NWDY
LDAY=ICDAT(LDYWFL+LDAY)
LDNM=LDAYNM+(LDAY-1)*8-1
WRITE(SYSOUT,2) (ICDAT(LDNM+I),I=1,8),CDAT(LDAY+CPDOFF),
1 CDAT(LDAY+SPDOFF),TOURNM

FIND NEXT TOUR. CALCULATE AVERAGE CALL RATE,
SERVICE TIME, EFFECTIVE CARS

LTOUR=0
LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
IF(LTOUR EQ 0) GO TO 40
LTMN=LTRNM+(ITYPE-1)*8-1
IF(ICODE(LTOUR+TYYTOFF) NE 5) GO TO 55
WRITE(SYSOUT,3) (ICDAT(LTMN+I),I=1,8),CDAT(LTOUR+ACTOFF)
GO TO 40
ISTART=ICODE(LTRST+ITYPE-1)
IEND=ICODE(LTRED+ITYPE-1)
ST=0.
DO 60 I=ISTART,IEND
ST=ST+CDAT(LDAY+STDOFF+I-1)
TOURLN=IEND-ISTART+1
ST=ST*60./TOURLN
CR=CDAT(LTOUR+CRTOFF)/TOURLN
EF=0.
DO 70 IBLK=1,2
IBDT=ICODE(LRTTB(IBLK)+ITYPE-1)
IF(IBDT LT 1) GO TO 70
IBRD=ICODE(LBRFL+IBDT-1)
LBLK=LDAY+BLDOFF+(IBRD-1)*NWD
BLKLT=ICODE(LBLKTB(2)+IBDT-1)-ICODE(LBLKTB(1)+IBDT-1)+1
EF=EF+BLKLN*CDAT(LBLK+EFDOFF)
CONTINUE
EF=EF/TOURLN
WRITE(SYSOUT,3) (ICDAT(LTNN+I),I=1,8),CDAT(LTOUR+ACTOFF),
1 CDAT(LTOUR+VTOFF),,ST,CR,(CDAT(LTOUR+HFTOFF+I-1),I=1,3)
GO TO 50
FORM OF PRECINCT HEADER
FORMAT(1H,8A1,2H:,8A1,,; AREA=*,F5.1,,; STREET MILES=*,
1 F5.1,,; B2=*,F5.3,,; B1=*,F5.3)

FORMAT FOR DAY HEADER AND COLUMN LABELS
FORM(,, DAY=*,8A1,, CALL RATE PARM=*,F5.2,, SERVICE TIME*,
1341
1  \text{* PARM='F5.2//21X,}
2  AVG.  AVG.  AVG.  FRAC.  FRAC.  FRAC.'/16X,  \text{1343}
3  ACT.  EFF.  RSP.  PTL.  SERV  CALL  OF  P1  OF  P2  OF  P3'\)
3  6X,8A1, 2X,  \text{1344}
4  'CARS  CARS  VEL.  VEL.  TIME  RATE  CALLS  CALLS  CALLS')
 \text{1345}
\text{FORMAT  FOR  ENTRIES  IN  COLUMNS}
\text{1346}
\text{FORMAT(6X,8A1,6(2X,F4.1),3(2X,F5.3))}
\text{1347}
\text{END}
\text{1348}
\text{1349}
\text{1350}
Subroutine SETWFL

Subroutine SETWFL (set work flags) is called to set the "work" flags (LTRWFL, LDYWFL) described in Table 2 after a command has been successfully interpreted. Division flags (LDIVFL) are also set.

These flags define the subsets of days, tours, and divisions (among those that have been read) which will be operated on by the current command. These subsets are determined from the phrases of the command qualifier. The qualifier phrases must have been converted to name lists by subroutine GTDSPC before SETWFL is called. For days and tours, each work flag corresponds to one day or tour that has been read. If a day or tour is selected by a command qualifier, then the value of its work flag will be the relative position of the day or tour among all the days or tours in the data base; otherwise, its value will be zero. If no day or tour names appear in a command qualifier, then all days or tours read are implicitly selected; otherwise, only those named are selected. Names that do not appear in the data base are ignored.

Division flags (in LDIVFL) correspond to names of divisions in a list produced by the READ command (LDIVNM). Flags of divisions named in the command qualifier are set to one (1); others are set to zero. Division flags are referenced by a division number associated with each precinct. The condition of no division names in the command qualifier is detected in subroutine NXPCT.

The parameter IERR is set to 1 if any errors are detected in SETWFL; otherwise, its value on return will be zero.
SUBROUTINE SETWFL(IERR)

SET WORK FLAGS
ASED ON QUALIFIER SCANNED BY GTDSPC

COMMON/SYSTEM/SYNS,SYNSOUT,IFILE,LIT
INTEGER SYNS,SYNSOUT

COMMON/KEYWDS/NKYWD,NTYPES,TYPDFF(4),KEYWID(8,30),WDTYP(30)
INTEGER TYPDFF,WDTYP
DIMENSION PCLSNM(8),DCLSNAME(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYWID(1,4)),(DCLSNAME,KEYWID(1,3)),
1(TOURNAM,KEYWID(1,2))

COMMON/STORE/TOP,BOT,ROBOT,MABOT,NWRODS,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)

COMMON/PNTRN/LOVRLY,LOVTR(2),
1(PCTD,TNPCTR,LMPCTD,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2(LDFRL,NDAYRD,LDYFRL,NTRDT,LTRTB(2),LSTRT,LTREND,LTFRL,LTRNM,
3(NTRRD,LTFWFL,NBLDT,LBLKTB(2),LBRLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
4(LDIVNM,LDIVFL

SET DAY FLAGS

IERR=0
IF(NDAYRD .LT. 1) GO TO 105
DO 10 I=1,NDAYRD
ICDAT(LDFRL+I-1)=0
LN=LNMLST(1)
N=NNAMES(1)
IF(N .NE. 0) GO TO 30
IDAY=0
10 CONTINUE
DO 20 I=1,NDAYDT
IF(ICDAT(LDFRL+I-1) .EQ. 0) GO TO 20
IDAY=IDAY+1
ICDAT(LDFRL+IDAY-1)=1
CONTINUE
GO TO 100
IF(N .EQ. 0) GO TO 100
I=LKP8(ICDAT(LNM),ICDAT(LDAYMN),NDAYDT)
IF(I .EQ. 0) GO TO 40
IDAY=ICDAT(LDFRL+I-1)
IF(IDC .LT. 1) GO TO 40
ICDAT(LDFRL+IDAY-1)=1
LN=LNML+8
N=N-1
GO TO 30

SET TOUR FLAGS

IF(NTRRD .GT. 0) GO TO 108
WRITE(SYSOUT,1)
FORMAT('** NO PRIOR READ COMMAND - REENTER.**')
IERR=1
RETURN
DO 110 I=1,NTRRD
ICDAT(LTRWFL+I-1)=0
LNM=LNMLST(2)
N=NNAMES(2)
IF(N .NE. 0) GO TO 130
ITOUR=0
DO 120 I=1,NTRDT
IF(ICDAT(LTRRFL+I-1) .EQ. 0) GO TO 120
ITOUR=ITOUR+1
ICDAT(LTRWFL+ITOUR-1)=I
CONTINUE
GO TO 200
IF(N .EQ. 0) GO TO 200
I=LKP8(ICDAT(LNM),ICDAT(LTRNM),NTRDT)
IF(I .EQ. 0) GO TO 140
ITOUR=ICDAT(LTRRFL+I-1)
IF(ITOUR .LT. 1) GO TO 140
ICDAT(LTRWFL+ITOUR-1)=I
LNM=LNM+8
N=N-1
GO TO 130

SET DIVISION FLAGS

IF(NDIVRD .EQ. 0) RETURN
DO 210 I=1,NDIVRD
ICDAT(LDIVFL+I-1)=0
LNM=LNMLST(3)
N=NNAMES(3)
IF(N .EQ. 0) RETURN
I=LKP8(ICDAT(LNM),ICDAT(LDIVNM),NDIVRD)
IF(I .EQ. 0) GO TO 230
ICDAT(LDIVFL+I-1)=1
LNM=LNM+8
N=N-1
GO TO 220
END
Function NXPCT

Function NXPCT (next precinct) is called during command execution to determine the next precinct selected by a command qualifier. Its argument (LPCT) is a pointer to the data for a precinct in array DATA. On entry, LPCT points to the last precinct selected (zero if none) and the value of the function is a pointer to the next precinct selected (zero if none).

A precinct is selected if any of the following criteria are met:

- No precinct or division names appear in the command qualifier
- The entry in table LDIVFL (see Chapter IV) corresponding to the precinct's division is nonzero
- The precinct's name appears in the PRECINCT phrase of the command qualifier.
FUNCTION NXPCT(LPCT)
FINDS THE NEXT PRECINCT SELECTED AFTER LPCT

COMMON/STOR,E/TP,TOP,BOT,ROBOT,MAXBOT,NWDBTS,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICAT(11000)
EQUIVALENCE (ICAT,CDAT)

COMMON/OFFST/NMPOFF,DVPOFF,APPOFF,SMPOFF,B1POFF,B2POFF,DYPFF,
1 NWDPC,T,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
2 QUTOFF,QXTOFF,CTOFF,QTOFF,QTTOFF,CTOFF,ITYTOFF,ACTOFF,RVTOFF,
3 VTOFF,HTOFF,MTOFF,LETTOFF,PRRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
4 EFOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBFF,CTBFF,NWDBL
INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPFF,CPDOFF,
1 SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QTDFF,QXTOFF,CTOFF,QTTOFF,
2 QTOFF,CTOFF,ITYTOFF,ACTOFF,RVTOFF,PVTOFF,HTOFF,BLDOFF,
3 EFOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBFF,CTBFF,QOBOFF,QNBOFF

COMMON/PNTRS/IVCRLY,IVCTR(2),
1 NPDCTDT,NPCTR,LPACTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2 LDYRF,NDYRD,LDYWFL,NTRDT,LRD(2),LRTST,LTRMTN,LTDDLF,LTRNL,
3 NTRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVD,LNIVRD,
4 LDVNLNM,LDVFL

NXPCT=LPCT
IF(LPCT .NE. 0) GO TO 10
NXPCT=LPACTDT
GO TO 20
NXPCT=NXPCT+NWDPT
IF((NXPCT .LE. LPACTDT)+(NPCTR-1)*NWDPT) GO TO 30
NXPCT=0
RETURN
IF(NNAMES(3)+NNAMES(4) .EQ. 0) RETURN
IF(NNAMES(3) .EQ. 0) GO TO 40
IDIV=ICAT(NXPCT+DVPFF)
IF(ICAT(LDVNL+IDIV-1) .NE. 0) RETURN
IF(NNAMES(4) .EQ. 0) GO TO 10
I=LKBP(ICAT(NXPCT+NMPFF),ICAT(LNMLST(4)),NNAMES(4))
IF(I .EQ. 0) GO TO 10
RETURN
END
Function NXDAY

Function NXDAY (next day) is used to index through the selected days for a precinct during command execution. Its arguments LPCT and LDAY are pointers to the data for a precinct and to the data for the last day selected for the precinct, respectively (see Chapter IV). The value of the function is a pointer to the next day selected after LDAY (zero if none). On entry, a value of zero for LDAY indicates that the first day selected for the precinct is to be located.

A day is selected if and only if the value of its corresponding work flag is nonzero (see the discussion of Subroutine SETWFL).

FUNCTION NXDAY(LPCT, LDAY)
...}
FINDS THE NEXT DAY SELECTED IN PRECINCT LPCT AFTER LDAY
...}
COMMON/STORE/TOP, BOT, RDBOT, MAXBOT, NWORDS, CDAT(11000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICAT(11000)
EQUIVALENCE (ICAT, CDAT)
...
COMMON/PNTRS/I0VRLY, I0VTR(2),
1NPCTOT, NPCTRD, LPCTOT, LNMNST(4), NNAMES(4), NDAYDT, LDAYNM,
2LDYRF, NDAYR, LDYWFL, NTRDT, LTRRB(2), LTRST, LTREND, LTRRFL, LTRNM,
3NTRRD, LTRRF, NBLDT, LBLKTB(2), LBLRF, NBLND, LBLWFL, NDIVG, NDIVRD,
4LDIVNM, LDIVFL
...
COMMON/OFFSET/NMPOFF, DVPOFF, ARPFF, S4POFF, 31POFF, B2POFF, DYPOFF,
1NDOFF, CPDOFF, SPDOFF, VDOFF, CRDOFF, STDOFF, TRDOFF, NWDDY,
2QTOFF, QXTOFF, CRTOFF, QTOFF, QNTOFF, CTTOFF, TYTOFF, ACTOFF, RVTTOFF,
3PVTOFF, HFTOFF, MFTOFF, LTTOFF, NPRIO, NWDT, BLDOFF, QBOFF, QNBOFF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, NWDBL
...
INTEGER DVPFF, ARPFF, S4POFF, 31POFF, B2POFF, DYPFF,
1SPDOFF, VDOFF, CRDOFF, STDOFF, TRDOFF, QTOFF, QXTOFF, CRTOFF, QTOFF,
2QNTOFF, CTTOFF, TYTOFF, ACTOFF, RVTTOFF, PVTOFF, HFTOFF, BLDOFF,
3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, QBOFF, QNBOFF
...
NXDAY=LDAY
IF(LDAY .NE. 0) GO TO 10
NXDAY=LPCT+DYPFF
GO TO 20
0
NXDAY=NXDAY+NWDDY
0
IDAY=(NXDAY-LPCT-DYPFF)/NWDDY+1
IF(IDAY .LE. NDAYRD) GO TO 30
NXDAY=0
RETURN
0
IF(ICAT(LDYWFL+IDAY-1) .EQ. 0) GO TO 10
RETURN
END
Function NXTOUR

Function NXTOUR (next tour) is used to index through the selected tours of a day during command execution. Its arguments LDAY and LTOUR are pointers to the data for a day within a precinct and the last tour selected within the day, respectively. The value of the function is a pointer to the next tour selected after LTOUR (zero if none). On entry, a value of zero for LTOUR indicates that the first tour selected for the day is to be located.

A tour is selected if and only if the value of its corresponding work flag is nonzero and the tour type is not equal to 1 (a tour type of 1 indicates that the tour holds a place that would be occupied by an overlay tour, but there is no overlay tour for the specified day).

On return, the parameter ITYPE is set to the value of the tour work flag. This value is the relative position of the tour among all the tours in the data base and is useful for referencing the tables that contain tour starting and ending times and mappings of tours to blocks.
FUNCTION NXTOUR(LDAY, LTUR, ITYPE)
FINDS THE NEXT TOUR SELECTED IN LDAY AFTER LTUR.
ITYPE IS THE VALUE OF THE TOUR WORK FLAG.

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

COMMON/OFFSET/NMPOFF,DPPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPoff,
1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TDROFF,NWDDY,
2QTOFF,QXTOFF,CTTOFF,QOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,HFTOFF,LFTOFF,NPRIO,NWDTL,BLDOFF,QBQOFF,QNB0FF,
4EFBOFF,ACBOFF,AWBOFF,BRBOFF,RMBOFF,FCB0FF,CTBOFF,NWD8L

INTEGER DPPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPoff,CPDOFF,
1SPDOFF,OVDOFF,CRDOFF,STDOFF,TDROFF,QTOFF,CTTOFF,QOFF,
2QTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
3PVTOFF,HFTOFF,LFTOFF,NPRIO,NWDTL,BLDOFF,QBQOFF,QNB0FF

COMMON/PNTRS/I0VRLY,IOVTR(2),
1NPCRT,NPCTD,LPTCDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRF,NDAYRD,LDYWL,NTRTD,LTOTO(2),LTSRT,LTREND,LTRRFL,LTRNM,
3NTRD,LTRWFL,NBLDT,LBLKTB(2),LBRLFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
4LDMVNM,LDIVF

NXTOUR=LTUR
IF(NXTOUR .NE. 0) GO TO 10
NXTOUR=LDAY+TDROFF
GO TO 20
NXTOUR=NXTOUR+NWDTL
ITOUR=(NXTOUR-LDAY+TDROFF)/NWDTL+1
IF(ITOUR .LE. NTRRD) GO TO 30
NXTOUR=0
RETURN
ITYPE=ICDAT(LTRWFL+ITOUR-1)
IF(ITYPE .EQ. 0 .OR. ICDAT(NXTQJR+TYTOFF) .EQ. 1) GO TO 10
RETURN
END
Subroutine DISP

Subroutine DISP implements the DISP command. Its function is to display selected output tables for selected shifts.

DISP calls SCAN twice to get the user's table specification. Then GTDSPC is called to scan the command qualifier and SETWFL is called to determine the subset of shifts for which output will be displayed. Subroutine NRGORD determines the output order (in DORDER) that results from the previously established output order (RORDER) and the order of qualifier phrases in the current command (ORDER). We consider that the six possible permutations of the values in DORDER define six different ways of printing tables. The permutations are mapped onto unique integers by multiplying the elements of DORDER by successive powers of two. Table 10 gives the output orderings and their corresponding integer labels (after the labels have been transformed into successive integers in the range 1-6).

Table 10

<table>
<thead>
<tr>
<th>Integer Label</th>
<th>Output Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Precinct, Tour, Day</td>
</tr>
<tr>
<td>2</td>
<td>Tour, Precinct, Day</td>
</tr>
<tr>
<td>3</td>
<td>Precinct, Day, Tour</td>
</tr>
<tr>
<td>4</td>
<td>Day, Precinct, Tour</td>
</tr>
<tr>
<td>5</td>
<td>Tour, Day, Precinct</td>
</tr>
<tr>
<td>6</td>
<td>Day, Tour, Precinct</td>
</tr>
</tbody>
</table>

The integer labels are used as entries into a branch table that controls calls to routines to implement table displays in the various output orderings. In the program documented in this report, only the Day, Tour, Precinct and Precinct, Day, Tour orderings are implemented. The branch table is entered once for each table number specified by the user.

In the program documented here, only Tables 1 and 2 are valid. The subroutine provides flexibility for the user to implement other output orders and/or output tables.
SUBROUTINE DISP

IMPLEMENTS THE DISP COMMAND

COMMON/STATS1T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
INTEGER PORDER,RORDER

COMMON/SYSTEM/SYSIN,SYSDOUT,IFILE,LIT
INTEGER SYSIN,SYSDOUT

COMMON/KEYWDS/NKWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER TYPOFF,WDTYPE
DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYWD(I,4)),(DCLSNM,KEYWD(I,3))
1(TOURNM,KEYWD(I,2))

COMMON/SCODES/SEND,CMD,NULST,NAMLIST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

COMMON/STORE/TOP,BOT,ROBOT,MAXBOT,NWORDˆ,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)

DIMENSION ORDER(3),VAL(2),DORDER(3)
INTEGER ORDER,TYPE,VAL,DORDER

LGETT=TOP
TYPE=CMD

FINDS WHICH TABLE(S) ARE TO BE DISPLAYED

CALL SCAN(TYPE,VAL)
IF(TYPE .EQ. FSPEC) GO TO 20
WRITE(SYSDOUT,1)
FORMAT(10)**INVALID TABLE SPECIFICATION - REENTER.**
10 TOP=LGETT
RETURN
KEYVAL=VAL(1)
I=KEYVAL-TYPOFF(FSPEC)
IF(I .NE. 3) GO TO 10
CALL SCAN(TYPE,VAL)
IF(TYPE .NE. NULST) GO TO 10
NPARM=VAL(1)
LPARM=VAL(2)

INTERPRET QUALIFIER

CALL SCAN(TYPE,VAL)
CALL GTDSPC(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 30
TOP=LGETT
RETURN

SET WORK FLAGS

CALL SETWFL(IERR)
IF(IERR .EQ. 0) GO TO 35
TOP=LGETT
RETURN

SET OUTPUT ORDER

CALL MRGORD(ORDER,RORDER,DORDER)
IORD=0
DO 40 I=1,3
K=2**(I-1)
IORD=IORD*K*DORDER(I)
IF(IORD .GT. 14) IORD=IORD-1
IORD=IORD-10
DO 700 I=1,NPARAM
ITAB=ICDAT(LPAMP+(I-1)*2)
IF(ITAB .LT. 1 .OR. ITAB .GT. 2) GO TO 700

CALL ROUTINE TO DISPLAY TABLE

GO TO (100,200,300,400,500,600),IORD

CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
WRITE(SYSOUT,2)
FORMAT(1H /1H )
TOP=LGETT
RETURN
END
Subroutine DSPDPT

Subroutine DSPDPT (display by precinct, day, tour) is called by subroutine DISP to print DISP command output tables by precinct, day, and tour. Its parameter ITAB specifies the table number to be printed.

Subroutine ZERO is called to initialize accumulators for averages at the overall, precinct, day, and tour levels. The integer parameter of ZERO specifies the level of the accumulators to be initialized, with one (1) corresponding to the highest level (overall) and four (4) corresponding to the lowest level (tour).

Functions NXPCT, NXDAY, and NXTOUR are used to index through the precincts, days, and tours selected by the user. A labeling line and column headings for tours are printed for each precinct and day displayed.

For each tour selected, a flag (FLAG) is set which contains the character to be printed at the left of each line of table output. Another flag, IADD (which is a parameter for subroutine COMPTB), indicates whether or not the tour is an overlay tour, and therefore whether its measures (level 4) are to be accumulated into the measures for a day (level 3). This may seem redundant here, but the decision to include overlay tour measures in higher levels of aggregation or not must be made at different levels for different output orders. Subroutine COMPTB is called to compute the measures of table ITAB for the tour. Subroutine PRTEB (print table) prints a line of output measures (at level 4, the tour level). Subroutine TOTAL adds measures for a specified level into the accumulators for the next highest level. TOTAL also computes and prints averages for all except level 4 measures. Statistics are not printed for a level of aggregation if there is only one entry for the next lower level.
SUBROUTINE DSPPD(T ITAB)

DISPLAYS TABLE ITAB IN ORDER OF TOUR WITHIN DAY WITHIN PRECINCT

COMMON/ SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT

COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,1NWDPCT,CPDPOFF,SPDPOFF,ODPOFF,CRDPOFF,STDPOFF,TRDPOFF,NWDDY
2QTOFF,QXTOFF,CRTOFF,QTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,3PVTOFF,HFTOFF,MHTOFF,HTOFF,NPRIO,NWDTA,BLDOFF,QBOFF,QNOFF,4EFBOFF,ACBOFF,AWBBOFF,CRBOFF,RMBOFF,OCBBOFF,CTBOFF,NWDBL
 INTEGER DVPFF,ARPOFF,SMPFF,B1POFF,B2POFF,DYPFF,CPDPOFF,1NWDPCT,CPDCT,SPDPOFF,ODPOFF,CRDPOFF,STDPOFF,TRDPOFF,QTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,3PVTOFF,HFTOFF,MHTOFF,HTOFF,NPRIO,NWDTA,BLDOFF,QBOFF,QNOFF

COMMON/PNTR/S,IQVRLY,IOVTR(2),1NPCTD,PNPCTD,LPCDT,LOCMDT,4NNAMES(4),5NDAYDT,LDAYNM,6LDYFRL,6DAYS,7DNRTD,8RTB(2),9LTST,10TREN,NLRTMLE,LRTMLE,123NLRTD,4NRTLMF,5BLDLT,6BLKTB,7BLRLF,8INNERD,9NDIVND,10NDIVLD,11LDYPLY,12LCNLD
 COMMON/STORE/1TOP,2BOT,3RDBOT,4MAXBOT,5NWDBDS,6CDAT(11000)
 INTEGER TOP,BOT,RD BOT
 DIMENSION IDC DAT(11000)
 EQUIVALENCE(IDC DAT,CDAT)

COMMON/KEYWDS/NKWTD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
 INTEGER TYPOFF,WDTYPE
 DIMENSION PCLSINM(8),DCLSIN(8),TOURNM(8)
 EQUIVALENCE (PCLSINM,KEYWD(1,4)),(DCLSINM,KEYWD(1,3)),1(TOURNM,KEYWD(1,2))

DATA BLANK/1H/,*STAR/1H*/,PLUS/1H*/
 CALL ZERO(1)

FIND PRECINCT

NPCT=0
LPC T=0
LPCT=NLPCT(LPCT)
IF(LPCT,.EQ.,0) GO TO 100
NPCT=NPCT+1
NDAY=0
CALL ZERO(2)
LDAY=0

FIND DAY

LDAY=NXDAY(LPCT,LDAY)
IF(LDAY,.EQ.,0) GO TO 80
NDAY=NDAY+1
IDAY=(LDAY-LPCT-DYPOFF)/NWDDY
LDAY=ICDAT(LDYFRL+IDAY)

1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
LDAYNM=LDAYNM+(IDADY-1)*8-1
WRITE(SYSOUT,1) PCLSNNM,(ICDAT(LPCT+NMPOFF+I-1),I=1,8),1
1 (ICDAT(LDNM+I),I=1,8)

FORMAT CFR PRECINCT AND DAY HEADER
FORMAT(*' ',8A1,' : ' ,8A1,': DAY: ',8A1)
CALL TITLE(ITAB,TOURNM)
NTOUR=0
LTOUR=0
CALL ZERO(3)

FIND TOUR
LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
IF(LTOUR .EQ. 0) GO TO 60
IND=ICDAT(LTOUR+TYTOFF)
FLAG=BLANK
IF(IND .LT. 3) GO TO 40
FLAG=STAR
IF(IND .EQ. 5) FLAG=PLUS
NTOUR=NTOUR+
CALL ZERO(4)
IADD=1
IF(IND .EQ. 5) IADD=0

COMPUTE OUTPUT MEASURES
CALL COMPTB(ITAB,LPCT,LDAY,LTOUR,ITYPE,IADD)

PRINT OUTPUT MEASURES FOR SHIFT
CALL PRTBL(ITAB,4,FLAG,ICDAT(LTRNM+(ITYPE-1)*8))
GO TO 30

ACCUMULATE MEASURES FOR DAYS
CALL TOTAL(ITAB,3,NTOUR,1)
GO TO 20
IF(NDAY .LT. 1) RETURN
IF(NDAY .GT. 1)WRITE(SYSOUT,2) PCLSNNM,
11ICDAT(LPCT+NMPOFF+I-1),I=1,8)

FORMAT FOR PRECINCT HEADER
FORMAT(*' ',8A1,' : ' ,8A1)

ACCUMULATE MEASURES FOR PRECINCTS
CALL TOTAL(ITAB,2,NDAY,1)
GO TO 10
IF(NPCT .LT. 2) RETURN
WRITE(SYSOUT,3)
FORMAT(*' GRAND')
CALL TOTAL(ITAB,1,NPCT,0)
RETURN
END
Subroutine DSPDTP

Subroutine DSPDTP controls DISP command output when the output order is precinct, within tour, within day. Parameter ITAB specifies the output table that is to be displayed.

DSPDTP operates in a manner similar to that of DSPDPDT. The primary differences are in the meanings of the different levels of aggregation and in the way the next shift to be displayed is found. The routines NXPCT, NXDAY, and NXTOUR are set up to vary the tour most quickly, followed by the day and precinct. This corresponds exactly to the output order of tour within day within precinct, but not to precinct within tour within day. Therefore, for each day selected, DSPDTP determines the number of times that NXDAY will have to be called after a precinct is located to get to the day. DSPDTP also computes the number of times that NXTOUR must be called to get to a particular tour after a precinct and day have been located.
SUBROUTINE DSPDTP(ITAB)

DISPLAYS TABLE ITAB IN ORDER OF PRECINCT WITHIN TOUR WITHIN DAY

COMMON/STORE/Top, BDT, RDBOT, MAXBOT, NWDRDS, CDAT(11000)
INTEGER TOP, BDT, RDBOT
DIMENSION ICAT(11000)
EQUIVALENCE (ICAT, CDAT)

COMMON/OFFSET/NMPCT, DVPCT, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPCT, 1
NWDRCT, CPDCT, SPDCT, OVDCT, CRDOFF, STDOFF, TRDOFF, NWDDY,
2QTOFF, QXTOFF, CTQTOFF, QTQTOFF, QNTOFF, CTNTOFF, QTQTOFF, ACTOFF, RVTOFF,
3PVTQTOFF, HPFTOFF, MFTOFF, LFTOFF, NPRINT, NWDRT, BLDOFF, QBOFF, QNBUFF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBUFF, CTBOFF, NWDBL

INTEGER DVPCT, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPCT, CPDCT, 1
SPDCT, OVDCT, CRDOFF, STDOFF, TRDOFF, QTOFF, QXTOFF, CTQTOFF, QTQTOFF, 2
QNTOFF, CTNTOFF, QTQTOFF, ACTOFF, RVTOFF, PVTQTOFF, HPFTOFF, MFTOFF, LFTOFF, BLDOFF,
3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBUFF, CTBOFF, QBOFF, QNBUFF

COMMON/PNTR/SYSIN, SYSDT, IFIL, LIT
INTEGER SYSIN, SYSDT

COMMON/KEYWD/KEYW(64), NTYPES, TYPQTOFF(4), KEYW(8, 30), WDTTYPE(30)
INTEGER TYPQTOFF, WDTTYPE
DIMENSION PLCNSTM(8), DCLSNM(8), TOURNM(8)
EQUIVALENCE (PLCNSTM, KEYW(1, 4)), (DCLSNM, KEYW(1, 3)), 1
TOURNM, KEYW(1, 2)

DATA BLANK/1H ,STAR/1H**,PLUS/1H**/

NDAY=0
NXTDAY=0
CALL ZERO(1)

FIND POSITION OF NEXT DAY AMONG SELECTED DAYS

DO 15 I=1, NDAYRND
15 IF(IDAY.LT. NXTDAY) GO TO 20
CONTINUE
GO TO 100
NXTDAY=IDAY
NDAY=NDAY+1
LDNM=LADYMN+(IDAY—1)*8—1
CALL ZERO(2)

FIND TYPE OF NEXT DAY

NXTYPE=0
NTOUR=0
DO 40 I=1,NTRRD
I_TYPE=ICDAT(LTRWFL+I-1)
IF(I_TYPE.GT.NTYPE) GO TO 50
CONTINUE
IF(INTOUR.GT.1) WRITE(SYSOUT,2) (ICDAT(LDNM+I),I=1,8)
   FORMAT FOR DAY HEADER
FORMAT(/' DAY: ',8A1)
   ACCUMULATE MEASURES FOR DAY
CALL TOTAL(ITAB,2,NTOUR,1)
GO TO 10
NTYPE=I_TYPE
NTOUR=NTOUR+1
CALL ZERO(3)
LTNM=LTNM+(I_TYPE-1)*8-1
WRITE(SYSOUT,1) (ICDAT(LDNM+I),I=1,8),TOURNM,(ICDAT(LTNM+I),I=1,8)
   FORMAT OF DAY AND TOUR HEADER
FORMAT(/' DAY: ',8A1,';',2(1X,8A1))
CALL TITLE(ITAB,PLCSNM)

   FIND NEXT PRECINCT
LPCT=0
NPCT=0
LPCT=NXPCT(LPCT)
IF(LPCT.NE.0) GO TO 65
IADD=1
IF(IOVRLY.EQ.1 .AND. NTYPE.EQ.4TRDT) IADD=0

   ACCUMULATE MEASURES FOR TOUR
CALL TOTAL(ITAB,3,NPCT,IADD)
GO TO 30

   GET DAY
LDAY=0
DO 70 I=1,NDAY
LDAY=NXDAY(LPCT,LDAY)

   GET TOUR
LTOUR=0
LTOUR=NXTOUR(LDAY,LTOUR,I_TYPE)
IF(LTOUR.EQ.0) GO TO 60
IF(I_TYPE.NE. NTYPE) GO TO 80
NPCT=NPCT+1
FLAG=BLANK
IND=ICDAT(LTOUR+TYTOFF)
IF(IND.LT.3) GO TO 90
FLAG=STAR
IF(IND.EQ.5) FLAG=PLUS
CALL ZERO(4)

   COMPUTE AND PRINT MEASURES
CALL COMPTB(ITAB,LPCT,LDAY,LTOUR,I_TYPE,1)
CALL PRTBL(ITAB,4,FLAG,ICDAT(LPCT*NMP*OFF))
GO TO 60
IF(NDAY .LT. 2) RETURN
WRITE(SYSOUT,4)
FORMAT(' GRAND')
CALL TOTAL(ITAB,1,NDAY,0)
RETURN
END
Subroutine ZERO

Subroutine ZERO is called by the subroutines that control table output to clear level N accumulators. ZERO initializes array T, which is used to accumulate weighted sums for output measures, and array S, which is used to accumulate weights.

```
SUBROUTINE ZERO(N)
C
C INITIALIZE ACCUMULATORS FOR LEVEL 'N' TABLE OUTPUT
C
COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
INTEGER PORDER,RORDER
C
DO 10 I=1,8
  S(N,I)=0.
  T(N,I)=0.
10  RETURN
END
```

1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
Subroutine TITLE

Subroutine TITLE is called by the routines that control DISP command table output to print column headings. Parameter ITAB specifies the table for which headings are to be printed. NAME is an array that contains eight characters in Al format used as a heading to identify the leftmost column. NAME can identify the column entries as being tour names, precinct names, or day names (although the day name option has not been implemented).

SUBROUTINE TITLE(ITAB, NAME)

PRINTS COLUMN HEADINGS FOR TABLE ITAB

COMMON SYSTEM/SYSIN, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

DIMENSION NAME(8)

GO TO (10, 20), ITAB

0 WRITE(SYSOUT, 1) NAME
  FORMAT(/, 12X,
  1'AVG. AVG. AVG. PATROL AVG. AV PTL FREQ'/12X,
  2'UTIL. UTIL. TRAV. HRS PER PATROL TIMES SUPP AVG CARS'/
  3 2X, 8A1, 2X,
  4'(EFF) (ACT) TIME SUPP CR FREQ. CR PER HR AVAIL.')
RETURN

0 WRITE(SYSOUT, 2) NAME
  FORMAT(/, 12X,
  1'ACT. CAR CALL SERV PROB CALL AVG P2 AVG P3 AVG TOT'
  1/2X, 8A1, 2X,
  2'CARS HRS RATE TIME DELAYED DELAY DELAY DELAY')
RETURN
END

1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
Subroutine COMPTB

Subroutine COMPTB (compute table) is called from the routines that control DISP command output to compute output measures for one shift. Parameter ITAB specifies the table for which measures are to be computed. LPCT, LDAY, and LTOUR are pointers to the data for the precinct, day, and tour to be used in the computation. ITYPE is the relative position of the tour among all tours in the data base; it provides an index to tour starting and ending times. IAWD indicates whether or not the measures computed for the tour are to be included in the next higher level of aggregation (this depends on the DISP command output order and on whether or not the shift is an overlay).

For either output table, weighted sums and weights are computed for all measures and summed over all blocks of the shift. The measures are either computed directly from data items in CDAT or by function references to such routines as AVTT for average travel time or OBJFl for fraction of calls delayed. Weighted sums and weights are accumulated in the columns of row 4 of arrays T and S, respectively. If requested, the contents of row 4 of arrays T and S are added to the contents of row 3; this represents inclusion of the measures for the shift in the next higher level of aggregation. Finally, averages of the measures over the blocks of the shift are computed by dividing the weighted sums in T by the weights in S. Array CIND is set to print an asterisk next to the value of the limiting constraint (if any).
SUBROUTINE COMPTB(ITAB, LPCT, LDAY, LTOUR, ITYPE, IADD)

COMPUTES ONE OUTPUT LINE OF ONE TABLE

COMMON/OFFSET/NMPOFF, DVPDIFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYP3FF, 1NWDCT, CPDQFF, SPDOFF, OVDIFF, CRDOFF, STDOFF, TRDOFF, NWODY,
2QTOFF, QXTOFF, CRTOFF, QTOFF, CTOFF, QTTOFF, ACTOFF, RVTOFF, 3PVENT, HFTOFF, MFTOFF, LFTOFF, NPRIO, NWOTR, BLOFF, QBOFF, QNB0FF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, CCBOFF, CTBOFF, NWDBL

INTEGER DVPDIFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYP3FF, CPDQFF, 1SPDOFF, OVDIFF, CRDOFF, STDOFF, TRDOFF, NWODY,
2QTOFF, QXTOFF, CRTOFF, QTOFF, CTOFF, QTTOFF, ACTOFF, RVTOFF, 3PVENT, HFTOFF, MFTOFF, LFTOFF, NPRIO, NWOTR, BLOFF, QBOFF, QNB0FF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, CCBOFF, CTBOFF, NWDBL

COMMON/STORE/TOP, BOT, RDBOT, MAXBOT, NWORDS, CDAT(11000)

INTEGER TOP, BOT, RDBOT

DIMENSION ICDAT(11000)

EQUIVALENCE (ICDAT, CDAT)

COMMON/PNTSTS, TDYRLY, IQVTR(2),
1NPDCT, NPCTRD, LPCTDT, LNLMLT(4), NNMES(4), NDAYDT, LDAYNM,
2LDYRFL, NDAYRD, LDYWFL, NTQRT, LRTRB(2), LRST, LTRRFL, LTRNM,
3NTRRD, LTRWF, NBLDT, LBLKTB(2), LBLRFL, NBLRD, LBLWFL, NDIVDT, NDIVRD,
4LDIVMN, LDIVFL

COMMON/STATS/T(4, 8), S(4, 8), PORDER(3), RORDER(3), CIND(8)

INTEGER PORDER, RORDER

DATA BLANK//, STAR//

DIMENSION ICNSTR(10)

DATA ICNSTR(11/1, ICNSTR(2)/3, ICNSTR(3)/7, ICNSTR(4)/4/
1 ICNSTR(5)/5, ICNSTR(6)/1, ICNSTR(7)/5, ICNSTR(8)/6/
2 ICNSTR(9)/7, ICNSTR(10)/8/

LCT = LDAY + CRDOFF

LST = LDAY + STDOFF

IEND = ICDAT(LTREN = ITYPE - 1)

ISTART = ICDAT(LTRST = ITYPE - 1)

ILEN = IEND - ISTART + 1

TOURL = ILEN

GO TO (10, 500), ITAB

COMPUTE TABLE 1 MEASURES

DO 20 IBLK = 1, 2

IF( IBLD .GT. 1) GO TO 20

IBLKL = ICDAT(LTRTB(IBLK) + ITYPE - 1)

IF (IBLKL LT ITYPE) GO TO 20

IBLKL = ICDAT(LBLRFL + IBLD - 1)

LBD = LDAY + BLOFF * (IBLKL - 1) * NWDBL

IBEND = ICDAT(LBLKTB(2) + IBLD - 1)

IBSTRB = ICDAT(LBLKTB(1) + IBLD - 1)

BKLNL = IBEND - IBSTRB + 1

EF = CDAT(LBLK + EFB0FF)

IF OVERLAY TOUR, GET DATA FROM OVERLAID TOUR

LTOUR = LTOUR

20 CONTINUE
IF(ICDAT(LTOUT+TTYOFF) .EQ. 5)
1 LTTOUT=LDAY+TRDOFF+(IOVTR(IBLK)-1)*NWDOTR
RV=CDAT(LTOUT+RTVOFF)
AWL=CDAT(LBLK+AWDOFF)

UTILIZATION (EFFECTIVE)
X=AWL/EF
Y=BLKLN*EF
T(4,1)=T(4,1)+X*Y
S(4,1)=S(4,1)+Y

UTILIZATION (ACTUAL)
ACT=CDAT(LBLK+ACBDOFF)
X=AWL/ACT
Y=BLKLN*ACT
T(4,2)=T(4,2)+X*Y
S(4,2)=S(4,2)+Y

TRAVEL TIME
X=AVTT(IBSTRT, IBEND, LPCT, LDAY, RV, EF)
Y=CDAT(LBLK+CRBDOFF)
T(4,3)=T(4,3)+X*Y
S(4,3)=S(4,3)+Y

PATROL HOURS/ SUPP CRIME
X=BLKLN*(EF-AWL)
Y=CDAT(LBLK+OCBDOFF)
T(4,4)=T(4,4)+X
S(4,4)=S(4,4)+Y

PATROL FREQUENCY
T(4,5)=T(4,5)+X*CDAT(LTOUT+PVTDOFF)
S(4,5)=S(4,5)+BLKLN*CDAT(LPCT+SMPOFF)

PATROL FREQ * SUPP CR/HR
T(4,6)=T(4,6)+X*CDAT(LTOUT+PVTDOFF)*CDAT(LBLK+OCBOFF)
S(4,6)=S(4,6)+BLKLN**2*CDAT(LPCT+SMPOFF)

AVERAGE CARS AVAILABLE
X=EF-AWL
T(4,7)=T(4,7)+X*BLKLN
S(4,7)=S(4,7)+BLKLN
CONTINUE

ACCUMULATE MEASURES IF REQUESTED
IF(IADD .LT. 1) GO TO 40
DO 30 I=1,7
IF(S(4,I) .EQ. 0.) T(4,I)=0.
T(3,I)=T(3,I)+T(4,I)
S(3,I)=S(3,I)+S(4,I)

1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
CONTINUE

COMPUTE AVERAGES

DO 50 I=1,7
CIND(I)=BLANK
IF(S(4,I) .EQ. 0.) GO TO 50
T(4,I)=T(4,I)/S(4,I)
CONTINUE

SET CONSTRAINT INDICATOR

IC=ICDAT(LTOUR+CTTOFF)
IF(IC .GT. 5 .OR. IC .LT. 1) RETURN
CIND(INSTR(IC))=STAR
RETURN

COMPUTE TABLE 2 MEASURES

ACT=CDAT(LTOUR+ACTTOFF)
T(4,1)=ACT
S(4,1)=1.
T(4,2)=ACT*TOURLN
S(4,2)=1.
DO 520 IBLK=1,2
IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
IF(IBLD .LT. 1) GO TO 520
ILBR=ICDAT(LBLRFL+IBLD-1)
LBLK=IELD+BDTOFF*(IBLR-1)*NWDBL
ISTART=ICDAT(LBLKTB(1)+IBLD-1)
IEND=ICDAT(LBLKTB(2)+IBLD-1)
BLKLN=IEND-ISTART+1
LTOUR=LTOUR
IF(ICDAT(LTOUR+TYTOFF) .EQ. 5)
LTOUR=LDAY+HDTOFF*(IDVTR(IBLK)-1)*NWDIR
LFR=LTOUR+HFTOFF

CALL RATE

T(4,3)=T(4,3)+CDAT(LBLK+CRBOFF)
S(4,3)=S(4,3)+BLKLN

SERVICE TIME

ST=CDAT(LBLK+AWBOFF)*BLKLN*60.
T(4,4)=T(4,4)+ST
S(4,4)=S(4,4)+CDAT(LBLK+CRBOFF)

PROB. CALL DELAYED

EF=CDAT(LBLK+EFBOFF)
X=OBJF1(ISTART,IEND,LCR,LST,EF)
T(4,5)=T(4,5)+X
S(4,5)=S(4,5)+CDAT(LBLK+CRBOFF)

AVG P2 DELAY

X=OBJF2(2,ISTART,IEND,LCR,LST,LFR,EF)*60.
Y = CDAT(LBLK+CRBOFF) * CDAT(LFR+1)
T(4,6) = T(4,6) + X
S(4,6) = S(4,6) + Y

AVG P3 DELAY

X = OBJF2(3, ISTART, IEND, LCR, LST, LFR, EF) * 60.
Y = CDAT(LBLK + CRBOFF) * CDAT(LFR + 2)
T(4,7) = T(4,7) + X
S(4,7) = S(4,7) + Y

AVG TOTAL DELAY

RV = CDAT(LTTOUR + RVTOFF)
X = OBJF3(ISTART, IEND, LPCT, LDAY, RV, EF) * 60.
T(4,8) = T(4,8) + X
S(4,8) = S(4,8) + CDAT(LBLK + CRBOFF)
CONTINUE
N = 8
IF(IADD LT 1) N = 2
DO 530 I = 1, N
IF(S(4,I) EQ 0.) T(4,I) = 0.
T(3,I) = T(3,I) + T(4,I)
S(3,I) = S(3,I) + S(4,I)
DO 550 I = 1, 8
CIND(I) = BLANK
IF(S(4,I) EQ 0.) GO TO 550
T(4,I) = T(4,I) / S(4,I)
CONTINUE
IC = ICDAT(LTTOUR + CTTOFF)
IF(IC LT 6 OR IC GT 10) RETURN
CIND(IKSTR(IC)) = STAR
RETURN
END
Subroutine PRTBL

Subroutine PRTBL (print table) prints one line of Table 1 or Table 2 output.* The line can represent any level of aggregation from one shift to an overall average. Parameter ITAB specifies the table number to be printed and that implies the format and number of items to be written. LEV is the level of aggregation of statistics that are to be printed. PRTBL assumes that T(LEV,N) contains the output measure that will be printed in column N+1 of the line of output (this will have been computed by TOTAL or COMPTB, depending on LEV). NAME is an eight-character identifier that will be printed in the first output column. In the current version NAME can be a tour name, a precinct name, or the word "AVERAGE," depending on the output order and the level of aggregation. FLAG is a one-character indicator that is printed at the left of an output line to show the overlay status of a shift.

PRTBL assumes that CIND(N) contains a one-character indicator (asterisk or blank) that will be printed to the left of the (N+1)st column to indicate whether or not the corresponding measure was the limiting constraint in a MEET command.

*This refers to the two types of output that can be obtained using the DISP command, not to the tables in the present report.
SUBROUTINE PRTBL(ITAB,LEV,FLAG,NAME) 2139
PRINTS ONE LINE OF TABLE ITAB 2140
COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8) 2141
INTEGER PORDER,RORDER 2142
COMMON/SYSTEM/SYSIN,SYSSOUT,IFILE,LIT 2143
INTEGER SYSIN,SYSSOUT 2144
DIMENSION NAME(8) 2145
GO TO (10,20),ITAB 2146
0 WRITE(SYSSOUT,1) FLAG,NAME,(CIND(I),T(LEV,I),I=1,7) 2147
FORMAT(1H,A1,8A1,1X,A1,F4.3,1X,A1,F4.3,2X,A1,F4.1,1X,2(2X,A1 2148
1,F5.2),3X,A1,F6.3,3X,A1,F6.2) 2149
RETURN 2150
0 WRITE(SYSSOUT,2) FLAG,NAME,(CIND(I),T(LEV,I),I=1,8) 2151
FORMAT(1H,A1,8A1,1X,A1,F4.1,1X,A1,F5.1,1X,A1,F4.1,A1,F5.1,3X,A1 2152
1,F4.3,3X,3(1X,A1,F6.2)) 2153
RETURN 2154
END 2155
Subroutine TOTAL

Subroutine TOTAL is called from the routines that control DISP command output. Its function is to add weighted sums and weights for a specified level of output measures to the accumulators for the next higher level. Averages are computed for the specified level and printed via a call to PRTEL.

Parameter ITAB specifies the table of output measures being displayed. LEV specifies the level of measures to be printed. N gives the number of observations at level LEV+1 that are reflected in the level LEV sums (if N is less than 2, no level LEV statistics are printed). IADD indicates whether or not the accumulation of level LEV sums into LEV-1 is to take place (this depends on overlay considerations).
SUBROUTINE TOTAL(ITAB, LEV, N, IADD)

ACCUMULATES SUMS FOR WEIGHTED AVERAGES IN TABLES

COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
INTEGER PORDER, RORDER

COMMON/SYSTEM/SYSIN, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

DIMENSION AV(8)
DATA AV(1)/1HA/, AV(2)/1HV/, AV(3)/1HE/, AV(4)/1HR/, AV(5)/1HA/, AV(6)/1HG/, AV(7)/1HE/, AV(8)/1H/, FLAG/1H/

IF(N .LT. 1) RETURN
IF(LEV .LT. 2) GO TO 15
IF(ITAB .EQ. 1 .AND. IADD .EQ. 0) GO TO 15
M=8
IF(ITAB .EQ. 2 .AND. IADD .EQ. 0) M=2
LEV1=LEV-1
DO 10 I=1, M
T(LEV1,I)=T(LEV1,I)+T(LEV,I)
S(LEV1,I)=S(LEV1,I)+S(LEV,I)
10 CONTINUE
IF(S(LEV,I) .GT. 0.0) GO TO 16
T(LEV,I)=0.
GO TO 17
T(LEV,I)=T(LEV,I)/S(LEV,I)
CIND(I)=FLAG
IF(LEV .EQ. 4 .OR. N .LT. 2) RETURN
WRITE(SYSOUT,1)
FORMAT(1H1)
CALL PRTBL(ITAB,LEV,FLAG,AV)
GO TO (20,30), ITAB
RETURN

X=T(LEV,1)*S(LEV,1)
Y=T(LEV,2)*S(LEV,2)
WRITE(SYSOUT,3) X, Y
FORMAT(' TOTAL',3X,F6.1,1X,F6.1)
RETURN
END
Function AVTT

Function AVTT returns the average travel time to incidents over a specified span of hours of a particular day in a precinct. Parameters ISTART and IEND give the first and last hour for which travel time is computed. LPCT and LDAY are pointers to the data for the precinct and day. RV is the response speed of patrol units and EF is the number of effective cars on duty.

The formula used to determine the travel time for each hour is given in Appendix B of the User's Manual. The travel times for each hour are weighted by the number of calls in the hour. The travel time returned is in minutes.
FUNCTION AVTT(ISTART, IEND, LPCT, LDAY, RV, EF)

CALCULATES AVERAGE TRAVEL TIME

COMMON/STORE/TOPT, BOT, RDBOT, MAXBOT, NWDRS, CDAT(11000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

COMMON/OFFSET/DEMOFF, DVPOFF, ARPoff, SMPOFF, B1POFF, B2POFF, DPPOFF,
1NWPOFF, CDPOFF, SPDPOFF, CVPOFF, CRDPOFF, STDOFF, TRDOFF, NWDDY,
2QETOFF, QXTOFF, CRTOFF, QDTOFF, QDTOFF, CRTOFF, TYTOFF, ATOFF, RVTOFF,
3PVTOFF, HFTOFF, MFTOFF, LFTOFF, NPROD, NWTR, BLDOFF, QROFF, QBOFF, QNBOFF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, QCBOFF, CTBOFF, NWDBL

INTEGER DVPPOFF, ARPPOFF, SMPOFF, B1POFF, B2POFF, DPPOFF, CDPOFF,
1SPPOFF, OVPOFF, CRDPOFF, STDOFF, TRDOFF, QDTOFF, QDTOFF, CRTOFF, QDTOFF,
2QETOFF, CRTOFF, TYTOFF, ATOFF, RVTOFF, PVTOFF, HFTOFF, MFTOFF, LFTOFF, NPROD,
3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, QCBOFF, CTBOFF, QROFF, QBOFF, QNBOFF

TD=0.
CRT=0.
LCR=LDAY+CROFF-1
LST=LDAY+STDOFF-1
A=CDAT(LPCT+ARPPOFF)
STRONS=CDAT(LPCT+SMPOFF)/A
G=(STRONS-1.)/(STRONS-2.)
SRQTA=SQRT(G)
DO 30 I=ISTART, IEND
CR=CDAT(LCR+I)
ST=CDAT(LST+I)
CRT=CR+CR
AVAVL=EF-CR*ST

USE TRAVEL DISTANCE FUNCTION APPROPRIATE FOR AVG
CARS AVAILABLE IN AN HOUR

IF(AVAVL .GE. 1.) GO TO 10
TD=TD+.678*SRQTA*CR
GO TO 30
IF(AVAVL .GE. 2.) GO TO 20
TD=TD+SRQTA*(.08+.598/SQRT(AVAVL))*CR
GO TO 30
TD=TD+.711*CR*SRQTA/SQRT(AVAVL)
CONTINUE

COMPUTE AVERAGE TRAVEL TIME FROM TRAVEL DISTANCE WITH
STREET DENSITY CORRECTION

AVTT=60.*G*TD/(CRT*RV)
RETURN
END
Function OBJF1

Function OBJF1 (objective function 1) returns the weighted sum of the probability that a call will be delayed over a span of hours of a day in a precinct. It is called from COMPTB for Table 2 output, from KNSTR when constraints are being met, and from OBJFUN when car hours are being allocated. Parameters ISTART and IEND specify the span of hours. LCR and LST are pointers to the hourly call-rate and service-time data for the day, and EF is the number of effective cars on duty.

The probability that a call is delayed in each hour is given by PQQUEUE; this is weighted by the number of calls in the hour.

```
FUNCTION OBJF1(ISTART,IEND,LCR,LST,EF)                     2258
    : CALCULATES WEIGHTED SUM OF PROBABILITY THAT A CALL      2259
    : WILL BE DELAYED                                         2260
    :                                                        2261
    : COMMON/STORE/ TOP,BOT, RDBOT, MAXBOT, NWJRDS, CDAT(11000) 2262
    : INTEGER TOP, BOT, RDBOT                                  2263
    : DIMENSION ICDAT(11000)                                   2264
    : EQUIVALENCE (ICDAT, CDAT)                                2265
    :                                                         2266
    : CRT=0.                                                   2267
    : Q=0.                                                     2268
    : DO 10 I.HOUR=ISTART, I.END                               2269
    : CR=CDAT(LCR+I.HOUR-1)                                    2270
    : ST=CDAT(LST+I.HOUR-1)                                    2271
    : AWL=CR*ST                                                2272
    : Q=Q*PQUEUE(AWL, EF)*CR                                    2273
    : OBJF1=Q                                                  2274
    : RETURN                                                   2275
    : END                                                      2276
```

Function PQ\textsc{ueue}

Function PQ\textsc{ueue} (probability of queue) computes the probability that a call will be delayed, given the product of the service time and call rate (AWL) and number of effective cars (EF). It is invoked to obtain the probability of a call being queued before dispatch for one hour of a day in a precinct.

The formula used to compute PQ\textsc{ueue} is given in Appendix B of the User's Manual. If EF is not an integer, the value of PQ\textsc{ueue} is computed for the greatest integer less than EF and for the smallest integer greater than EF, and linear interpolation is used to obtain a function value for EF.

\begin{verbatim}
FUNCTION PQ\textsc{ueue}(AWL,EF)

CALCULATES PROBABILITY THAT A CALL WILL BE DELAYED

N=EF
XN=N
C=0.
SUM=1.
T=1.
IF(N .LT. 2) GO TO 110
NM1=N-1
DO 100 I=1,NM1
C=C+1.
T=T*AWL/C
SUM=SUM+T
0
C=C+1.
T=T*AWL/C
X=T/(1.-AWL/XN)
PQ\textsc{ueue}=X/(SUM+X)
IF(XN .EQ. EF)RETURN
SUM=SUM+T
C=C+1.
T=T*AWL/C
X=T/(1.-AWL/C)
PQ=PQ\textsc{ueue}-(PQ\textsc{ueue}-PQ)*(EF-XN)
RETURN
END
\end{verbatim}
Function OBJF2

Function OBJF2 (objective function 2) returns the weighted sum of the average time that a call of a specified priority can expect to wait before dispatch. The sum is taken over a span of hours of a day in a precinct. The delay is in hours.

Parameter N specifies the priority level of interest. ISTART and IEND specify the span of hours over which the weighted sum is to be taken. LST and LCR are pointers to the hourly service times and call rates for the day. LFR is a pointer to an array that contains the fraction of calls in each priority class for the shift in which the hours occur. EF is the number of effective cars on duty.

The formula used to compute the expected delay in an hour in a specified priority class is given in Appendix B of the User's Manual. The delay for each hour between ISTART and IEND is weighted by the number of calls in the priority class in the hour.
FUNCTION OBJF2(N, ISTART, IEND, LCR, LST, LFR, EF) 
COMPUTE WEIGHTED SUM OF PRIORITY N CALL DELAYS OVER 
OURS ISTART TO IEND 

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWDRDS,CDAT(11000) 
INTEGER TOP,BOT,RDBOT 
DIMENSION ICDAT(11000) 
EQUIVALENCE (ICDAT, CDAT) 

COMMON/OFFSET/NMPDFF,DVDPOFF,ARPPOFF,SMPOFF,B1PPOFF,B2PPOFF,DYPPOFF, 
1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY, 
2QTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF, 
3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPrio,NWDTR,BLDOFF,QDPOFF,QNBOFF, 
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBBOFF,CTBOFF,NWDBL 
INTEGER DVPPOFF,ARPPOFF,SMPOFF,B1PPOFF,B2PPOFF,DYPPOFF,CPDOFF, 
1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QOTOFF,QXTTOFF,CRTOFF,QNTOFF, 
2QTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF, 
3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBBOFF,CTBOFF,QDPOFF,QNBOFF 

COMMON/SYSTEM/SYSIN,SYSONT,IFILE,LIT 
INTEGER SYSIN,SYSONT 

IF(N.LT.1) GO TO 50  
CMFRN1=0.  
NM1=N-1  
FRN=CDAT(LFR+NM1)  
IF(NM1.LT.1) GO TO 20  
DO 10 I=1,NM1  
CMFRN1=CMFRN1+CDAT(LFR+I-1)  
CMFRN1=CMFRN1*FRN  
W=0.  
DO 30 I=ISTART,IEND  
CR=CDAT(LCR+I-1)  
ST=CDAT(LST+I-1)  
AWL=CR*ST  
ENMU=EF/ST  
Q=QUEUE(AWL,EF)  
W=W+CR*FRN*Q/(ENMU*(1.-CR*CMFRN1/ENMU)*(1.-CR*CMFRN1/ENMU))  
OBJF2=W  
RETURN 

COMPUTE AVERAGE DISPATCH DELAY FOR ALL CALLS 

W=0.  
DO 60 I=ISTART,IEND  
CR=CDAT(LCR+I-1)  
ST=CDAT(LST+I-1)  
AWL=CR*ST  
Q=QUEUE(AWL,EF)  
W=W+Q*CR/(EF/ST-CR)  
OBJF2=W  
RETURN 
END
Function OBJF3

Function OBJF3 (objective function 3) computes the weighted total delay (queuing + travel time) that a randomly selected call can expect to experience before a patrol car arrives, summed over a span of hours of a day in a precinct.

Parameters ISTART and IEND specify the span of hours over which the weighted sum is to be taken. LPCT and LDAY are pointers to the data for the precinct and day. RV is the response speed of patrol cars and EF is the number of effective cars on duty.

The formulas for computing travel time and expected delay in an hour are given in Appendix B of the User's Manual. The queuing delays and travel times are weighted by the number of calls in each hour.
FUNCTION OBJF3(ISTART, IEND, LPCT, LDAY, RV, EF)
COMPUTES WEIGHTED SUM OF TOTAL RESPONSE TIMES

COMMON/STORE/TOP, BOT, RDBOT, MAXBOT, NWORDS, CDAT (11000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

COMMON/OFFSET/NMPOFF, DVPOFF, ARPQFF, SPMQFF, B1POFF, B2POFF, DYPQFF,
1 NWDPCT, CPQOFF, SPDQFF, OVDQFF, CRDQFF, STDQFF, TRDQFF, NWDDY,
2 QDQFF, QXTOFF, CRTOFF, QTOFF, QNTOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF,
3 VTOFF, HFTOFF, MFTOFF, LTTOFF, NPRIQ, NWTR, BLDQFF, QBOFF, QNBOFF,
4 EBFQFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, QCBQFF, CTBOFF, NWDBL

INTEGER DVPQFF, ARPQFF, SPMQFF, B1POFF, B2POFF, DYPQFF, CPQOFF,
1 SPDQFF, OVDQFF, CRDQFF, STDQFF, TRDQFF, QTOFF, QNTOFF, CTTOFF, TYTOFF,
2 VTOFF, HFTOFF, MFTOFF, LTTOFF, NPRIQ, NWTR, BLDQFF, QBOFF, QNBOFF

TD=0.
W=0.
L=LDAY+CRDQFF-1
LST=LDAY+STDQFF-1
A=CDAT(LPCT+ARPQFF)
SRTA=SQRT(A)
DO 40 I=ISTART, IEND
CR=CDAT(LCR+I)
ST=CDAT(LST+I)
AWL=CR*ST
AVAVL=EF-AWL
IF(AVAVL .GE. 1.) GO TO 10
TD=.678*SQRTA*CR+TD
GO TO 40
IF(AVAVL .GE. 2.) GO TO 20
TD=TD+SRTA*(.08+.598/SQRT(AVAVL))*CR
GO TO 40
TD=(.711*SRTA/SQRT(AVAVL))*CR+TD
W=W+(QUEUE(AWL, EF)/(EF/ST-CR))*CR
STRDNS=CDAT(LPCT+SPMQFF)/A
G=(STRDNS-1.)/(STRDNS-2.)
TT=G*TD/RV
OBJF3=W+TT
RETURN
END
Subroutine SET

Subroutine SET implements the SET command. Its function is to alter the values of specified data items that have been read from the data base.

Successive calls to subroutine SCAN get pointers to lists of numbers that specify the types of data items to be altered and the values they are to assume. If the lists constitute a valid specification, subroutine GTDSPC is called to scan the command qualifier and SETWFL is called to set the "work" flags for the days and tours in the scope of the command.

In the main processing loop of SET, the program indexes through all selected precincts. For each precinct, the program indexes through all data item-value pairs specified by the user. If a data item applies to precincts as a whole (unavailability parameters are of this type), then the value of the data item for the precinct is changed to the specified value and the next data item-value pair is examined. If a data item applies to days within precincts (e.g., call-rate and service-time parameters) or to tours within days (e.g., actual cars assigned and response speed), then SET indexes through all selected days. If the data item applies to days as a whole, then the change is made for each day in turn. If the data item applies to tours, then the change is made to all selected tours within the day. If the user specifies that the number of suppressible crimes for a tour is to be changed, then proportional changes are made in the number of suppressible crimes in each block of the tour.

When all changes have been applied to all days for a precinct, subroutine DERIVE is called for each day to compute average workloads and effective cars for each block, and to insure that the resulting number of effective cars on duty in each block of each day is sufficient to handle the cfs workload.
SUBROUTINE SET

IMPLEMENTS THE SET COMMAND

COMMON/PNTRS/IOVRLY,IOVTR(2),
1NPCTDT, NPCTR0,LPCOYX,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRFL,NDAYRD,LDFYRFL,LNDTLY,LNTRB(2),LTRST,LTREND,LTRFLL,LTRNM,
3NRTRD,LTRWFL,NBLDT,LBLKTBI(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
4LDIVNM,LDIVFL

COMMON/SYSTEM/SYSIN,SYSDOUT,IFILE,LIT
INTEGER SYIN,SYSDOUT

COMMON/STORE/TOP,BOT,RGBOT,MAXB0T,NWINDOWS,CDAT(11000)
INTEGER TOP,BOT,RGBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)

COMMON/DKWDS/NKYWD NTYPES,KEYOFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER WPTYPEOFF,KEYWD
DIMENSION PCLSMT(8),CCLSMT(8),TDMIN(8)
EQUIVALENCE (PCLSMT,KEYWD(1,4)),(CCLSMT,KEYWD(1,3)),
1(TOURNKEY,KEYWD(1,2))

COMMON/OFFSET/NMPOFF,DPWFOFF,ARPOFF,SMMPOFF,1P0FF,2P0FF,3P0FF,
1NWPOFF,CPOFF,SPWFOFF,VOWFOFF,CR0FF,ST0FF,TR0FF,NWDDY,
2QTOFF,QTOFF,CRT0FF,QTOFF,QT0FF,CT0FF,CTT0FF,TY0FF,ACT0FF,RV0FF,
3PVT0FF,HPT0FF,MT0FF,LT0FF,NPRIG,ND0T3,BLD0FF,QBOFF,QNBOFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,MB0FF,OCBOFF,CT0FF,NW0BL

INTEGER DPF0FF,ARPOFF,SMMPOFF,1P0FF,2P0FF,3P0FF,
1SPD0FF,3PD0FF,CR0FF,ST0FF,TR0FF,QD1FF,QT0FF,CRT0FF,QT0FF,
2QTOFF,CTT0FF,TY0FF,ACT0FF,RV0FF,PV1FF,HPT0FF,LT0FF,
3EFBOFF,ACBOFF,AWBOFF,CRBOFF,MB0FF,OCBOFF,CTB0FF,QBOFF,QNBOFF

COMMON/SCODES/SEND,NUMLST,NAMLST,FSPEC,DSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

DIMENSION ORDER(3),VAR(2)
INTEGER TYPE,VAL
LGETT=TOP
TYPE=CMD
CALL SCAN(TYPE,VAL)
IF(TYPE .EQ. FSPEC) GO TO 20
WRITE(SYSDOUT,1)
FORMAT("*/ *** INVALID PARAMETER SPECIFICATION - REENTER")
TOP=LGETT
RETURN
KEYVAL=VAR(1)
I=KEYVAL-TYP0FF(FSPEC)
IF(I .NE. 1) GO TO 10

GET DATA TYPES, CHECK VALIDITY

CALL SCAN(TYPE,VAL)
IF(TYPE .NE. NUMLST) GO TO 10
NPARM=VAR(1)
LPARM=VAL(2)
DO 25 IPARM=1,NPARM
   I=ICDAT(LPARM+(IPARM-1)*2)
   IF(I.GT.0.AND.I.LT.11) GO TO 25
   WRITE(SYSOUT,4) I
   FORMAT(/' *** PARAMETER ***,I2,'"' INVALID - REENTER.'/) TOP=LGETT
RETURN
CONTINUE

GET DATA VALUES

CALL SCAN(TYPE,VAL)
IF(TYPE .EQ. NUMLST) GO TO 30
WRITE(SYSOUT,2)
FORMAT(/' *** INVALID PARAMETER VALUE - REENTER.'/) TOP=LGETT
RETURN
NVAL=VAL(1)
LVAL=VAL(2)
IF(NVAL .EQ. NPARM) GO TO 40
WRITE(SYSOUT,3)
FORMAT(/' *** NUMBER OF VALUES DOES NOT MATCH NUMBER OF PARMS' 1,:" - REENTER'/) TOP=LGETT
RETURN

SCAN QUALIFIER

CALL SCAN(TYPE,VAL)
CALL GTDSC(TYPE,VAL,ORDER)
IF(TYPE .EQ. ERR) GO TO 50
TOP=LGETT
RETURN

SET WORK FLAGS

CALL SETWFL(IERR)
IF(IERR .EQ. 0) GO TO 55
TOP=LGETT
RETURN
LPCT=0

GET NEXT PRECINCT

LPCT=NXPCT(LPCT)
IF(LPCT .EQ. 0) GO TO 140

LOOK AT DATA TYPES

DO 130 IPARM=1,NPARM
   NP=ICDAT(LPARM+(IPARM-1)*2)
   IF(NP .GT. 2) GO TO 70
   CDAT(LPCT+SMPOFF+NP)=CDAT(LVAL+(IPARM-1)*2+1)
GO TO 130

DAY-SPECIFIC DATA
LDAY=0
LDAY=NXDAY(LPCT,LDAY)
IF(LDAY .EQ. 0) GO TO 130
IF(NP .GT. 4) GO TO 90
N = NP-3
XPARM = CDAT(LVAL+(IPARM-1)*2+1)
RATIO = XPARM/CDAT(LDAY+N)
CDAT(LDAY+N) = XPARM
L1 = LDAY+GROFF
L2 = L1+23
DO 80 L = L1, L2
CDAT(L) = CDAT(L)*RATIO
GO TO 75

TOUR-SPECIFIC DATA

LTOUR = 0
LTUR = NATOUR(LDAY, LTOUR, ITYPE)
IF(LTOUR .EQ. 0) GO TO 75
IF(NP .EQ. 10) GO TO 100
N = NP-5
CDAT(LTOUR+N)*ACTOFF = CDAT(LVAL+(IPARM-1)*2+1)
GO TO 95

IBL1 = ICAT(LTRTB1+1)+ITYPE-1
IBL2 = ICAT(LTRTB2+1)+ITYPE-1
IBL3 = ICAT(LBRLFL+IBL1-1)
IF(IWB2 .NE. 0) IBL2 = ICAT(LBRLFL+IBL2-1)
LWK1 = LDAY+BLDOFF+(IBL1-1)*NWDBL
IF(IWB2 .NE. 0) LWK2 = LDAY+BLDOFF+(IBL2-1)*NWDBL
IF(IWB2 .NE. 0) CRM = CRM+CDAT(LWK2+OCB0FF)
XPARM = CDAT(LVAL+(IPARM-1)*2+1)
RATIO = XPARM/CRM
CDAT(LWK1+OCB0FF) = CDAT(LWK1+OCB0FF)*RATIO
IF(IWB2 .NE. 0) CDAT(LWK2+OCB0FF) = CDAT(LWK2+OCB0FF)*RATIO
GO TO 95

CONTINUE

RE-DERIVE BLOCK VALUES FOR EACH DAY AND CHECK FOR MINIMUM

DO 135 IDAY = 1, NDAYRD
LDAY = LPCT*DYOFF+(IDAY-1)*NWDDY
CALL DERIVE(LPCT, LDAY)
GO TO 60
CONTINUE
RETURN
END
Subroutine MEET

Subroutine MEET implements the MEET command. Its function is to assign enough cars to all shifts within its scope so that a user-specified set of constraints on selected performance measures is met.

At entry, successive calls to subroutine SCAN get pointers to the list of output measures (LPARM) and the list of constraint values (LVAL). If all of the output measure specifications are valid and the number of constraint values matches the number of output measure specifications, GTDSPC is called to scan the MEET command qualifier. Subroutine SETWFL sets the "work" flags for days and tours and subroutine CKOVR (check overlay) insures that if an overlay tour has been specified in the qualifier, then the overlaid tours have also been specified.

MEET then indexes through all selected precincts, days, and tours. The blocks of each shift thus selected are dealt with independently. If a block has not been within the scope of a previous MEET, ADD, or ALOC command since the last READ command (IDATA(LBLK+CTBOFF) less than 0), enough cars are assigned to the block to keep the utilization of an effective car under 1 in each of its hours.

Starting with either the current assignment or the minimum assignment, the number of cars in a block is increased as necessary to meet each specified constraint in turn. Function KNSTR determines whether or not a particular constraint has been met by a given number of effective cars (KNSTR is not used for minimum manning level--constraint 6).

When constraints have been met for all blocks of all tours of a day, subroutine STRCAR (set tour cars) is called to obtain a feasible allocation of cars to the tours of the day that will result in the required number of cars in each block (see Chapter III). Then SBLACT (set block actual cars) is called to convert this tour allocation to a block allocation (see Chapter III) and SBLEF (set block effective cars) is called to determine the resulting number of effective cars in each block.

MEET returns when all constraints have been met for all blocks of all shifts within the scope of the command.
SUBROUTINE MEET

DETERMINES CAR REQUIREMENTS TO MEET SPECIFIED CONSTRAINTS.

COMMON/STORE/TOP,BOT, RDBOT, MAXBOT, NWORDS, CDAT(11000)
INTEGER TOP, BOT, RDBOT
DIMENSION IC(DAT(11000))
EQUIVALENCE (ICD, CDAT)

COMMON/PNTRS/IOVRLY, IOVR(2),
1NPCTOT, NPLCDT, LPCTOT, LMLIST(4), NNAME(S(4), NDAYDT, LDAYNM,
2LDYRF, NDAYRD, LDYWF, INTRDT, LTRTB(2), LTSTR, LTREN, LTRRF, LTRNM,
3NTRRD, LTRWF, NBLDT, LBLKTB(2), LBLRF, NBLRD, LBLWF, NDIVDT, NDIVRD,
4LDIVNM, LDIVFL

COMMON/OFFSET/NMPOFF, DPOFF, ARPOFF, SPOFF, BIPOFF, B2POFF, DPOFF, DYPOFF,
1NWPCT, CPDOFF, SPD0FF, DDOFF, CRDOFF, TDOFF, NWDY,
2CDTOFF, QXTOFF, CRTOFF, QTOFF, QNTOFF, CTTOFF, TTOFF, ACTOFF, RTTOFF,
3PVTOFF, HPTOFF, NPTOFF, LPTOFF, NPROG, NWDTR, BLOGF, QBOFF, QNOFF,
4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, NWDOL

INTEGER DPOFF, ARPOFF, SPOFF, BIPOFF, B2POFF, DPOFF, CPDOFF,
1SPDOFF, DDOFF, CRDOFF, TDOFF, QTOFF, QNTOFF, PVTOFF, CRTOFF, QXTOFF,
2TTOFF, ACTOFF, RVTOFF, PVTOFF, HPTOFF, BLOGF,
3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCB0FF, CTBOFF, QBOFF, QNOFF

COMMON/IMPORT/SYSIN, SYSOUT, IFILE, LIT
INTEGER SYSIN, SYSOUT

COMMON/KEYWS/NKYSW, NTYPS, TYP0FF(4), KEYWD(8, 30), WDTYP(30)
INTEGER TYP0FF, WDTYP
DIMENSION PCLS(8), DCLS(8), TORN(8)
EQUIVALENCE (PCLS, KEYWD(1,4)), (DCLS, KEYWD(1,3))
1(TORN, KEYWD(1,2))

COMMON/SODES/SEND, CMD, NUMLST, NAMLST, FSPEC, DSPEC, DUM, ERR
INTEGER SEND, CMD, FSPEC, DSPEC, DUM, ERR

INTEGER TYPE, VAL
DIMENSION VAL(2)
LGETT=TOP
TYPE=CMD

GET CONSTRAINT SPECIFICATIONS, CHECK VALIDITY

CALL SCAN(TYPE, VAL)
IF(TYPE = EQ. FSPEC) GO TO 20
WRITE(SYSOUT, 1)
FORMAT('/ *** INVALID CONSTRAINT SPECIFICATION - REENTER')
TOP=LGETT
RETURN
KEYVAL=VAL(1)
I=KEYVAL-TYP0FF(FSPEC)
IF(I .NE. 2) GO TO 10
CALL SCAN(TYPE, VAL)
IF(TYPE .NE. NUMLST) GO TO 10
NPARM=VAL(1)
LPARM=VAL(2)
DO 25 IPARM=1,NPARM
I=ICDAT(LPARM+(IPARM-1)*2)
IF(I.GT.0.AND. I.LT.11) GO TO 25
WRITE(SYSOUT,2) I
FORMAT(/' *** INVALID CONSTRAINT NUMBER : ',I4,' - REENTER\')
TOP=LGETT
RETURN
CONTINUE

GET CONSTRAINT VALUES

CALL SCAN(TYPE,VAL)
IF(TYPE.EQ. NUMLST) GO TO 30
WRITE(SYSOUT,3)
FORMAT(/' *** INVALID CONSTRAINT VALUE(S) - REENTER\')
TOP=LGETT
RETURN
NVAL=VAL(1)
LVAL=VAL(2)
IF(NVAL.EQ. NPARM) GO TO 40
WRITE(SYSOUT,4)
FORMAT(/' *** NUMBER OF VALUES DOES NOT MATCH NUMBER OF CONS',L1^)' - REENTER\')
TOP=LGETT
RETURN

SCAN QUALIFIER

CALL SCAN(TYPE,VAL)
CALL GTDSPC(TYPE,VAL,ORDER)
IF(TYPE.NE. ERR) GO TO 50
TOP=LGETT
RETURN

SET WORK FLAGS

CALL SETWFL(IERR)
IF(IERR.EQ. 0) GO TO 55
TOP=LGETT
RETURN

INSURE THAT OVERLAY SEGMENT IS COMPLETE OR NOT INCLUDED

CALL CKOVRI(IERR)
IF(IERR.EQ. 0) GO TO 60
TOP=LGETT
RETURN
LPCT=0
NTOT=0
LPCT=NXPCT(LPCT)
IF(LPCT.NE. 0) GO TO 110
WRITE(SYSOUT,6) NTOT
FORMAT(/' ',I4,' CAR HOURS ALLOCATED\')
TOP=LGETT
RETURN
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
LDAY=0
120 LDAY=NXDAY(LPCT,LDAY)
   IF(LDAY.EQ.0) GO TO 100
   LTOUR=0
140 LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
   IF(LTOUR.NE.0 .AND. IDCAT(LTOUR+TYOFF).NE. 5) GO TO 160
   CALL STRCAR(LDAY,CARHRS)
   NTOT=NTOT+CARHRS
   CALL SBLACT(LPCT,LDAY)
   CALL SBLRF(LPCT,LDAY)
   GO TO 120
160 DO 220 IBLK=1,2
   IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
   IF(IBLD.EQ.0) GO TO 220
   IBLR=ICDAT(LBLRFL+2+IBLD-1)
   LBLK=LDAY+BLDOFF+(IBLR-1)*NWBL
   AW=CDAT(LBLK+AWBOFF)
   IF(ICDAT(LBLK+CTBOFF).LT.0) GO TO 165
   EF=CDAT(LBLK+EFBOFF)
   ACT=CDAT(LBLK+ACB0FF)
   GO TO 170
   C
   DETERMINE MINIMUM BLOCK REQUIREMENTS
   C
   165 EF= INT(CDAT(LBLK+RMBOFF)+1.)
   ACT=CEIL((EF*B1*AWL)/(1.-B2))
   CDAT(LBLK+ACB0FF)=ACT
   EF=ACT*(1.-(B1*AWL/ACT)+B2)
   CDAT(LBLK+EFBOFF)=EF
   IDCAT(LBLK+CTBOFF)=0
   C
   INSURE THAT ALL CONSTRAINTS ARE MET FOR EACH BLOCK
   C
   170 DO 200 IPARM=1,NPARM
   IP=ICDAT(IPARM+(IPARM-1)*2)
   CVAL=CDAT(LVAL+(IPARM-1)*2+1)
   IF(IP.NE.6) GO TO 180
   ACT=CVAL
   EF=ACT*(1.-(B1*AWL/ACT)+B2)
   GO TO 190
180 I=KNSTR(IP,CVAL,EF,LPCT,LDAY,LTOUR,LBLK,IBLD)
   IF(I.NE.0) GO TO 190
   ACT=ACT+1.
   EF=ACT*(1.-(B1*AWL/ACT)+B2)
   GO TO 180
190 IF(ACT.LE.CDAT(LBLK+ACB0FF)) GO TO 200
   CDAT(LBLK+ACB0FF)=ACT
   CDAT(LBLK+EFBOFF)=EF
   IDCAT(LBLK+CTBOFF)=IP
   CONTINUE
220 CONTINUE
GO TO 140
END
Subroutine STRCAR

Subroutine STRCAR (get tour cars) determines a feasible allocation of cars to the tours of a day so that the resulting number of cars in each block of the day will be at least as great as the number currently assigned. The number of cars currently assigned to each block of the day is the number required to meet some constraint and is set by MEET or ADDALC. Parameter LDAY is a pointer to the data for the day for which the tour assignment is to be determined. CARHRS, on return, is the total number of car hours that have been assigned to all tours of the day. The algorithm used to generate the assignment of cars to tours is given in Chapter III.
SUBROUTINE STRCARS(LDAY, CARHRS)

C DETERMINES FEASIBLE ALLOCATION OF CARS TO TOURS IN A DAY, GIVEN THE CAR REQUIREMENTS IN THE BLOCKS OF A DAY.

C COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICAT(11000)
EQUVALENCE (ICAT,CDAT)

C COMMON/PNTRS/IOVR,LOVR(2),
LNPDAT,NPDAT,LPDCT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRF,NDAYRD,LDYRFNLDT,TRB(2),LRTST,LRTSER,TRLRFL,TRLNM,
3NTRRD,TRLWFL,NBDT,LBLKTB(2),BLRRLF,NBLRD,MLRFN,NDIVOT,NDIVRD,
4LDIVNM,LDIVFL

C COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPPOFF,
LDNPCT,CPDOFF,SPDOFF,DVDOFF,CRDOFF,STDFF,TRDOFF,NWDY,
2QDTOFF,QXTOFF,CTOFF,QDTOFF,QDTOFF,CTOFF,CTOFTOFF,ACTOFF,RTDOFF,
3PVTOFF,HPTOFF,MTTOFF,LSTOFF,NPDQ,NDTR,DLOFF,QQOFOFF,QQNOFF,
4EBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,QCBOFF,CTBOFF,NWDL

C INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPPOFF,CPDOFF,
LSPDOFF,OVDOFF,CRDOFF,STDFF,TRDOFF,QUDFOFF,QXTOFF,CTOFF,QDTOFF,
2QDTOFF,CTOFF,CTOFTOFF,ACTOFF,RTDFOFF,PTOFF,HTOFF,DLOFF,
3EBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,QCBOFF,CTBOFF,QNOFF

C DIMENSION LBLK(2)

C CARHRS=0.
LTOUR=0
5 LTOUR=NXTOUR(LDAY, LTOUR, IYPE)
IF(LTOUR .EQ. 0) RETURN
ISTART=ICAT(LRTST+IYPE-1)
IEND=ICAT(LRTSER+IYPE-1)
TOURLN=IEND-ISTART+1

C GET POINTERS TO BLOCKS

C DO 10 IB=1,2
LBLK(IB)=0
IBLK=ICAT(LTRST(IB)+IYPE-1)
IF(IBLK .EQ. 0) GO TO 10
IBLK=ICAT(LBLRFL+IBLK-1)
LBLK(IB)=LDAY+BDLOFF+(IBLK-1)*NWDBL
10 CONTINUE
ID=ICAT(LTOUR+TTOFF)
GO TO (100, 20, 30, 40, 50), ID

C TOUR NOT IN OVERLAY SEGMENT

C 20 I=1
IF(LBLK(IB) .EQ. 0) GO TO 25
IF(CDAT(LBLK(IB)+ACBOFF) .LT. CDAT(LBLK(IB)+ACBOFF)) I=2
CDAT(LTOUR+ACTOFF)=CDAT(LBLK(IB)+ACBOFF)
CARHRS=CARHRS+TOURLN*CDAT(LTOUR+ACTOFF)
20 CONTINUE
I=(ICAT(LBLK(IB)+CTOFF)-ICAT(LBLK(IB)+CTBFOFF))
GO TO 5

FIRST OVERLAID TOUR

30 X1=CDAT(LBLK(1)+ACB0FF)
CDAT(LTOUR+ACTOFF)=X1
CARHRS=CARHRS+TOURLN*X1
ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(1)+CTBOFF)
X2=CDAT(LBLK(2)+ACB0FF)
GO TO 5

SECOND OVERLAID TOUR

40 X3=CDAT(LBLK(1)+ACB0FF)
X4=CDAT(LBLK(2)+ACB0FF)
CDAT(LTOUR+ACTOFF)=X4
CARHRS=CARHRS+X4*TOURLN
ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(2)+CTBOFF)
GO TO 5

OVERLAY TOUR

50 CDAT(LTOUR+ACTOFF)=AMAX1(X2-X1,X3-X4,0.)
CARHRS=CARHRS+CDAT(LTOUR+ACTOFF)*TOURLN
I=1
IF(X3-X4 .GT. X2-X1) I=1
ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(I)+CTBOFF)
IF(X2-X1 .LE. 0. AND. X3-X4 .LE. 0.) ICDAT(LTOUR+CTTOFF)=0

ADJUST OVERLAY SEGMENT ASSIGNMENTS IF THE NUMBER OF CAR HOURS USED CAN BE REDUCED

LOVTR=LTOUR
ENOV=CDAT(LOVTR+ACTOFF)
IF(ENOV .EQ. 0.) RETURN
DELTA=AMAX1(X2-X1,0.)-AMAX1(X3-X4,0.)
IF(DELTA .EQ. 0.) RETURN
ISW=1
IF(DELTA .LT. 0.) ISW=2
DELTA=ABS(DELTA)
ITRRO=LOVTR(ISW)
ITYPE=ICDAT(LTRWFL+ITRRD-1)
ILEN=ICDAT(LTREND+ITYPE-1)-ICDAT(LTRST+ITYPE-1)+1
I0VLN=ICDAT(LTREND+NTDRT-1)-ICDAT(LTRST+NTDRT-1)+1
IF(I0VLN .GE. I0VLN) RETURN
LTOUR=LDAY+TROOFF+(ITRRO-1)*NWDBTR
LPCT=(ILDAY-LPCTDT)/NWDPC+LPCTDT
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
CDAT(LOVTR+ACTOFF)=CDAT(LOVTR+ACTOFF)-DELTA
CDAT(LTOUR+ACTOFF)=CDAT(LTOUR+ACTOFF)+DELTA
CARHRS=CARHRS-DELTA*(I0VLN-ILEN)
IBDT=ICDAT(LTRTB(ISW)+ITYPE-1)
IBRD=ICDAT(LBLRFL+IBDT-1)
LBLK=LDAY+BLOFF+IBRD-1)*NWDBL
ACT=CDAT(LBLOCK+ACB0FF)+DELTA
CDAT(LBLOCK+ACB0FF)=ACT
AWL=CDAT(LBLOCK+AWBOFF)
CDAT(LBLOCK+EBOffset)=ACT*{1.0 - ((B1*AWL/ACT)+B2)}
INV=Z/ISW
IBDT=ICDAT(LTRTB(INV)+NTROT-1)
IBRD=ICDAT(LBLRFL+IBDT-1)
LBLock=LDAY*BLDOFF+(IBRD-1)*NWDBL
ACT=CDAT(LBLOCK+ACBOFF)-DELTA
CDAT(LBLOCK+ACBOFF)=ACT
AWL=CDAT(LBLOCK+AWBOFF)
CDAT(LBLOCK+EBOffset)=ACT*{1.0 - ((B1*AWL/ACT)+B2)}
100
RETURN
END
Function KNSTR

Function KNSTR determines whether a given number of effective cars on duty in a block of a day results in a specified constraint on an output measure being met. A function value of one (1) is returned if the constraint is met, otherwise a value of zero (0) is returned. Parameter ICNSTR specifies the output measure whose value, with Ef effective cars, is to be tested against constraint value CVAL. The valid values of ICNSTR are the output measure specifications given in the MEET command description in Chapter III of the User's Manual. LPCT, LDAY, LTOUR, and LBLK are pointers to the data for the precinct, day, tour, and block for which the output measure is to be tested. IBLD is the relative position of the block among all blocks in the data base (e.g., the third block of a day).

The output measure specified by ICNSTR is evaluated for the block, given EF effective cars. Some output measures are computed directly from available data; others are computed by function references. The resulting measure is tested against CVAL and the value of KNSTR set according to the outcome.
FUNCTION KNSTR(ICNSTR, CVAL, EF, LPCT, LDAY, LTUR, LBLK, IBLD)

C DETERMINES WHETHER CONSTRAINT CVAL ON PERFORMANCE MEASURE
C ICNSTR IS MET BY EF EFFECTIVE CARS
C
COMMON/STORE/TOP, BOT, RDBOT, MAXBOT, NWJRDS, CDAT(11000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

COMMON/PNTR, IOVRLY, IOVTR(2),
1NPCTDT, NPCTR DT, LPCDT, LNMLST(4), Nnames(4), NDAYDT, LDAYNM,
2LDYRF, LDYRFD, LDYWFL, NTRDT, LTRTB(2), LTRST, LTREND, LTRRFL, LTRNM,
3NTRRD, LTRWFL, NLBDT, LBLKTB(2), LBLRF, NBLRD, LBLWFL, NDIVDT, NDIVRD,
4LDIVNM, LDIVFL

COMMON/OFFSET/NMPOFF, DVOFF, ARPoff, SPOFF, B1POFF, B2POFF, DYPOFF,
1NWDPCT, CPDOFF, SDPOFF, OVDOFF, CRDOFF, STD0FF, TRDOFF, NWDDY,
2QTOFF, QXTOFF, CRTOFF, QTOFF, QT0FF, CTTOFF, TYTOFF, ACTOFF, RVTOFF,
3PVT0FF, HTOFF, MPT0FF, LTOFF, NPRIQ, NWDR, BLDOFF, QBOFF, QNB0FF,
4EFBOFF, ACBOFF, AWBOFF, CB0FF, RMBOFF, OCBOFF, CTBOFF, NWDBL

INTEGER DVOFF, ARPoff, SPOFF, B1POFF, B2POFF, DYPOFF,
1SPDOFF, SDPOFF, CRDOFF, STD0FF, TRDOFF, QTOFF, QXT0FF, CRTOFF, QT0FF,
2QTOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF, PVTOFF, HTOFF, MPT0FF, LTOFF,
3EFBOFF, ACBOFF, AWBOFF, CB0FF, RMBOFF, OCBOFF, CTBOFF, QBOFF, QNB0FF

KNSTR=0
ISTART=ICDAT(LBLKTB(1)+IBLD-1)
IEND=ICDAT(LBLKTB(2)+IBLD-1)
LCR=LDAY+CRDOFF
LST=LDAY+STD0FF
BLKLN=IEND-ISTART+1
GO TO (100, 200, 300, 400, 500, 600, 700, 800, 900, 1000), ICNSTR

100 X=CDAT(LBLK*AWBOFF)/EF
IF (X .LE. CVAL) KNSTR=1
RETURN

C

200 RV=CDAT(LTUR+RVTOFF)
X=AVT(I(START, IEND, LCR, LST, EF), RV, EF)
IF (X .LE. CVAL) KNSTR=1
RETURN

300 X=EF-CDAT(LBLK*AWBOFF)
IF (X .GE. CVAL) KNSTR=1
RETURN

400 X=(EF-CDAT(LBLK*AWBOFF))*BLKLN/CDAT(LBLK*OCBOFF)
IF (X .GE. CVAL) KNSTR=1
RETURN

500 X=(EF-CDAT(LBLK*AWBOFF))*CDAT(LTUR+RVTOFF)/CDAT(LPCT*SMPOFF)
IF (X .GE. CVAL) KNSTR=1
RETURN

600 RETURN

700 X=OBJFL(ISTART, IEND, LCR, LST, EF)/CDAT(LBLK*CRBOFF)
IF (X .LE. CVAL) KNSTR=1
RETURN

800 N=2
GO TO 910
N=3
910   LFR=LTUOHL+HFTOFF
      X=60.*OBJF2(N,ISTART,IEND,LCR,LST,LFR,EF)/
      \  (CDAT(LBLK+CRBOFF)+CDAT(LFR+N-1))
      IF(X .LE. CVAL) KNSTR=1
      RETURN
1000  RV=CDAT(LTOUHL+RVTOFF)
      X=60.*OBJF3(ISTART,IEND,LPCT,LDAY,RV,EF)/CDAT(LBLK+CRBOFF)
      IF(X .LE. CVAL) KNSTR=1
      RETURN
      END
Subroutine CKOVR

Subroutine CKOVR (check overlay) is used to insure that if the
user has selected an overlay tour in a command qualifier, then he has
also selected the overlaid tours. Its parameter IERR is set to zero (0)
on return if a valid specification has been made, otherwise it is set
to one (1). The determination of validity is based on the "work" flags
of the tours involved. See the section on table pointers in Chapter IV
for a description of the flags and tables involved.

SUBROUTINE CKOVR(IERR)

Checks to insure that all tours in an overlay segment have been
selected in a command or that they have all been omitted

COMMON/PNTRS, IOVRLY, IOVTR(2),
1NPCDTS, NPCTRDS, LPICTDT, LNMLST(4), NNAME(4), NDAYDT, LDAYNM,
2LDYRFL, NDAYRD, LDYWFL, NTRYD, LTRTB(2), LTRST, LTRSTN, LTRRFL, LTRNM,
3NTRRD, LTRWFL, NBLDT, LBLKTB(2), LBLRLF, NBLRD, LBLWFL, NDIVDT, NDIVRD,
4LDIVNM, LDIVFL

COMMON/SYSTEM, SYSIN, SYSOUt, IFILe, LIT
INTEGER SYSIN, SYSOUt

COMMON/STORE, TOP, RDBOT, MAXBOT, NWORDS, CDAT(11000)
INTEGER TOP, RDBOT
DIMENSION ICAT(11000)
EQUIVALENCE (ICAT, CDAT)

IERR=0
IOV=ICAT(LTRWFL+NTLTD-1)
IF(IOV+IOV .EQ. 0 .OR. IOV .EQ. 0) RETURN
IOV=ICAT(LTRWFL+IOV-1)
IOV1=ICAT(LTRWFL+IOVTR(1)-1)
IOV2=ICAT(LTRWFL+IOVTR(2)-1)
IF(IOV+IOV1+IOV2 .EQ. 0) RETURN
IF(IOV+NE. 0 .AND. IOV .NE. 0 .AND. IOV2 .NE. 0) RETURN
WRITE(SYSOUT, 1)
FORMAT(1) *** INVALID OVERLAY T/JUR SPECIFICATION - REENTER."
IERR=1
RETURN
END
**Subroutine ADDALC**

Subroutine ADDALC (add and allocate) implements the ADD and ALLOCATE commands. Its parameter ISW determines which command is to be executed. If ISW is less than 2, then the ALOC command is executed; otherwise, the ADD command is executed.

Successive calls to subroutine SCAN get the user's specification of the number of car-hours; subroutine GTDSPC scans the command qualifier; and additional calls to SCAN get the user's objective function specification. The user's specification of the number of car-hours is saved as follows: NHOURS holds the numeric part of any specification; ISTAR is 1, 0, or -1, depending upon whether the user's expression is of the form *n, n, or n-* (* alone is equivalent to *0).

If an asterisk appears in the expression giving the number of car-hours, the number of car-hours currently allocated to all selected shifts is determined and the expression is evaluated to give a number of car-hours to be allocated or added.

The program then indexes through all selected precincts and days. If an ALOC command is being executed, each block of each selected tour of each day is assigned just enough cars to handle its cfs workload and subroutines STRCAR, SELACT, and SBLEF are called to get a feasible allocation of cars to tours and to translate the tour allocation back to a block allocation; this step is skipped for ADD commands. The objective function is evaluated for each selected block of a day via a call to SBLOBJ and for each selected tour of a day via a call to STROBJ. The constraint indicators for each block of selected tours are set to zero. Subroutine ADJUST is called for ALOC commands to ensure that the initial allocation results in the minimum objective function value for the number of car-hours assigned in each day.

After the objective function has been evaluated for all shifts, the number of car-hours that remain to be allocated is computed and subroutine ADDCAR is called to allocate that number of car-hours.
SUBROUTINE ADDALC(ISW)

C PERFORMS ADD OR ALLOCATE FUNCTION, DEPENDING ON
C THE VALUE OF 'ISW'.

C

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICOUNT(11000)
EQUIVALENCE (ICOUNT,CDAT)

C

COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
INTEGER SYSIN,SYSOUT

C

COMMON/PNTRS/IOVRLY,IOVTR(2),
1NPCDO,TNPCRD,LPCDOL,LMNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LSTRT,LTRRDL,LTRRFL,LTRNM,
3TRRDL,LTRRFL,NBLTD,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDVDT,NDVRD,
4LDIVNM,LDIVFL

C

COMMON/OFFSET/NMPOFF,DPVOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DPOFF,
1NWDOFF,CPDOFF,SPDOFF,OSDOFF,CSRDOFF,STDOFF,TRDOFF,NWDDY,
2QDOFF,QTDOFF,CRDOFF,QTDOFF,QTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,MTTOFF,LTTOFF,NPRIO,NWDTR,BLDOFF,QBDOFF,QNBOFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

C

INTEGER DPVOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DPOFF,CPDOFF,
1SPDOFF,OSDOFF,CSRDOFF,STDOFF,TRDOFF,OSDOFF,CRDOFF,QTDOFF,QTOFF,
2QTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QBDOFF,QNBOFF

C

COMMON/KEYWDS/NKYWD,NTYPES,TYPPOFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER TYPPOFF,WDTYPE
DIMENSION PCLSNA(8),DCLSNA(8),TOURNM(8)
EQUIVALENCE (PCLSNA,KEYWD(1,4)),(DCLSNA,KEYWD(1,3)),
1(TOURNM,KEYWD(1,2))

C

COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

C

DIMENSION VAL(2),ORDER(3),EN(4)
INTEGER TYPE,VAL

C

INTEGER CHARST,CHRMN
DATA CHARST/1H=/,CHARMN/1H=/

C

LGETT=TOP
TYPE=CMD
NHOURS=0
ISTAR=0

C

GET EXPRESSION FOR CAR HOURS TO ALLOCATE
C

CALL SCAN(TYPE,VAL)
IF(TYPE .EQ. NUMLST) GO TO 20
IF(TYPE .EQ. NAMLST) GO TO 15

10 WRITE(SYSOUT,1)

FORMAT(1x,3x,*** INVALID NUMBER OF CAR HOURS TO ALLOCATE - *,

1 'REENTER')
TOP=LGETT
RETURN
15 IF(ICDAT(VAL(2)) .NE. CHARST) GO TO 10
ISTAR=1
CALL SCAN(TYPE,VAL)
IF(TYPE .NE. NUMLST) GO TO 25
NHOURS=ICDAT(VAL(2))
IF(NHOURS .GT. 0) GO TO 10
CALL SCAN(TYPE,VAL)
GO TO 25
20 NHOURS=ICDAT(VAL(2))
CALL SCAN(TYPE,VAL)
IF(TYPE .NE. NAMLST) GO TO 25
IF(ICDAT(VAL(2)) .NE. CHARMN .OR. ICDAT(VAL(2)+1) .NE. CHARST) GO TO 10
ISTAR=-1
CALL SCAN(TYPE,VAL)
C C SCAN QUALIFIER
C
25 CALL GTDSPC(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 30
TOP=LGETT
RETURN
C C SCAN AND VALIDATE OBJECTIVE FUNCTION SPECIFICATION
C
30 IF(TYPE .EQ. FSPEC) GO TO 60
50 WRITE(SYSOUT,2)
2 FORMAT(*/***INVALID OBJECTIVE FUNCTION - REENTER*)
TOP=LGETT
RETURN
60 KEYOFF=VAL(1)
I=KEYOFF-TYPOFF(FSPEC)
IF(I .NE. 4) GO TO 50
CALL SCAN(TYPE,VAL)
IF(TYPE .NE. NUMLST) GO TO 50
NPARI=VAL(1)
LPARI=VAL(2)
IFNCTN=ICDAT(LPARI)
IF(IFNCTN .LT. 1 .OR. IFNCTN .GT. 3) GO TO 50
IF(IFNCTN .NE. 2) GO TO 130
IF(NPARI .GT. 1) GO TO 70
CALL GETTOP(4,LPARI)
ICDAT(LPARI)=2
ICDAT(LPARI+2)=0
GO TO 130
70 IPRIQ=ICDAT(LPARI+2)
IF(IPRIQ .GE. 0 .AND. IPRIQ .LE. NPRIQ) GO TO 130
WRITE(SYSOUT,3)
3 FORMAT(*/***INVALID OBJECTIVE FUNCTION PARAMETER(S) - REENTER*)
TOP=LGETT
RETURN
C C SET WORK FLAGS
C
130 CALL SETWFL(IERR)
IF(IERR .EQ. 0) GO TO 132
TOP=LGETT
RETURN

C CHECK OVERLAY SPECIFICATION

C CALL CKOVRI(IERR)
IF(IERR .EQ. 0) GO TO 135
TOP=LGETT
RETURN
NCRHRS=0

C

C ** TO ALLOW THE USER TO SPECIFY CARS TO ALLOCATE INSTEAD OF CAR HOURS
C ** A STATEMENT SHOULD BE ADDED HERE TO MULTIPLY 'NHOURS' BY THE LENGTH
C ** OF A TOUR; E.G. INSERT THE STATEMENT NHOURS=NHOURS*8 IF ALL TOURS
C ** ARE EIGHT HOURS IN LENGTH AND CARS (INSTEAD OF CAR HOURS)
C ** ARE TO BE ALLOCATED.

C IF(ISTAR .EQ. 0) GO TO 180

C FIND NUM OF CAR HOURS ALREADY ASSIGNED TO SELECTED SHIFTS

C CRHRS=NCRHRS
LPCT=0

C

C LPCT=NXPCT(LPCT)
IF(LPCT .EQ. 0) GO TO 170
LDAY=0

C

C LDAY=NXDAY(LPCT,LDAY)
IF(LDAY .EQ. 0) GO TO 140
LTOUR=0

C

C LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
IF(LTOUR .EQ. 0) GO TO 150
TOURLN=ICDAT(LTREND+ITYPE-1)-ICDAT(LTRST+ITYPE-1)*1
CRHRS=CRHRS+CDAT(LTOUR+ACTOFF)*TOURLN
GO TO 160

C

C COMPUTE CAR HOURS AVAILABLE TO ALLOCATE

C

C NCRHRS=INT(CRHRS*/5)
NHOURS=NHOURS+ISTAR*NCRHRS

C

C DETERMINE INITIAL ASSIGNMENT (ALOC ONLY) AND EVALUATE
C CORRESPONDING OBJECTIVE FUNCTION VALUES FOR ALL SHIFTS

C

C

C NTOF=0
LPCT=0

C

C LPCT=NXPCT(LPCT)
IF(LPCT .EQ. 0) GO TO 300
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
LDAY=0

C

C LDAY=NXDAY(LPCT,LDAY)
IF(LDAY .EQ. 0) GO TO 200
IF(ISW .GT. 1) GO TO 245

C

C FIND MINIMUM ASSIGNMENT FOR EACH BLOCK

C IBL=0
220  LTOUR=0  
    LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)  
    IF(LTOUR .EQ. 0) GO TO 240  
    IND=ICDAT(LTOUR+TYTOFF)  
    IF(IND .EQ. 5) GO TO 240  
    DO 230 IBLK=1,2  
        IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)  
        IF(IBDT .LT. 1) GO TO 230  
        IBDR=ICDAT(LBLRFL+IBDT-1)  
        LBLK=LDAY+BLDOFF*(IBDR-1)*NWDBL  
        AWL=CDAT(LBLK+AWBOFF)  
        EF=INT(CDAT(LBLK+RMBOFF)+1.)  
        ACT=CEIL((EF+BL*AWL)/(1.-B2))  
        EF=ACT*(1.-((B1*AWL/ACT)+B2))  
        CDAT(LBLK+ACBOFF)=ACT  
        CDAT(LBLK+EFBOFF)=EF  
        IF(IND .NE. 3 .AND. IND .NE. 4) GO TO 230  
        IBL=IBL+1  
        EN(IBL)=ACT  
    230    CONTINUE  
    GO TO 220  
    
C    FIND MINIMUM FEASIBLE TOUR ASSIGNMENT  
C    
240    CALL STRCAR(LDAY,CARHRS)  
    NTOE=NTOE+CARHRS  
    CALL SBLACT(LPCT,LDAY)  
    CALL SBEFL(LPCT,LDAY)  
    LTOUR=0  
250    LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)  
    IF(LTOUR .NE. 0) GO TO 255  
    IF(TOVRLY .EQ. 0 .OR. ICODAT(LDAY+OVDOFF) .EQ. 0 .OR. ISW .GT. 1) 3175  
C    GO TO 210  
    
C    ADJUST INITIAL TOUR ASSIGNMENT TO MINIMIZE  
C    OBJECTIVE FUNCTION  
C    
    X1=AMAX1(EN(2)-EN(1),0.)  
    X2=AMAX1(EN(3)-EN(4),0.)  
    DELTA=X1-X2  
    CALL ADJUST(LPARM,LPCT,LDAY,DELTA)  
    GO TO 210  
    
255    ICODAT(LTOUR+CTTOFF)=0  
C    COMPUTE INITIAL OBJECTIVE FUNCTION VALUES FOR BLOCKS  
C    
    DO 260 IBLK=1,2  
        IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)  
        IF(IBDT .LT. 1) GO TO 260  
        IBDR=ICDAT(LBLRFL+IBDT-1)  
        LBLK=LDAY+BLDOFF*(IBDR-1)*NWDBL  
        ICODAT(LBLK+CTBOFF)=0  
        IF(ICODAT(LTOUR+TYTOFF) .EQ. 5) GO TO 260  
        CDAT(LBLK+ACBOFF)=CDAT(LBLK+ACBOFF)-2.  
        CALL SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK,IBDT)  
        CALL SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK,IBDT)  
    260    CONTINUE  
    CALL STROBJ(LDAY,LTOUR,ITYPE)
GO TO 250

C
C ALLOCATE REMAINING CAR HOURS
C
300 IF(ISW .LT. 2) GO TO 305
NLEFT=NHOURS
IF(NLEFT .GT. 0) GO TO 310
RETURN
305 NLEFT=NHOURS-NTOT
IF(NLEFT .GE. 0) GO TO 310
WRITE(SYSDUT,4) NTOT
4 FORMAT(/' *** ',i5,' CAR HOURS AlLOCATED./')
TOP=LGETT
RETURN
310 CALL ADDCAR(NLEFT,LPARM)
TOP=LGETT
RETURN
END
Subroutine SBLOBJ

Subroutine SBLOBJ increases the number of cars assigned to a block by one and evaluates a specified objective function with one car more than the new assignment. Both the new current objective function value and the objective function value with an additional car are saved.

LPARM is a pointer to the parameter list that specifies the objective function to be evaluated. LPCT, LDAY, LTOUR, and LBLK are pointers to the precinct, day, tour, and block to be operated upon. IBT is the relative position of the type of block among blocks in the data base.

SUBROUTINE SBLOBJ(LPARM, LPCT, LDAY, LTOUR, LBLK, IBT)
C
C DETERMINES THE OBJECTIVE FUNCTION FOR A BLOCK WITH ONE
C MORE ACTUAL CAR THAN CURRENTLY ALLOCATED.
C
COMMON/PNTRS,IOVRLY,IOVTR(2),
1NPCTD,NPCTR,DLPCTD,LMNLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRT(2),LTRST,LTRND,LTRFL,LTRNM,
3NTRRD,LTRWFL,LBLD,LBLKTB(2),LBLRFL,NBRLD,LBLWFL,NDIVDT,NDIVRD,
4LDIVNM,LDIVFL
C
COMMON/OFFSET/NMPOFF,DVPOFF,ARP0FF,SMPOFF,B1POFF,B2POFF,DYPOFF,
1NWDPCT,CPDOFF,SPDOFF,QVDOFF,CROFF,STDOFF,TRDOFF,NWDDY,
2QTOFF,QXTOFF,CTOFF,QTOFF,QTFOFF,CFTOFF,CTOFF,ACTOFF,RTOFF,
3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDT,BLDOFF,QBOFF,QNOFF,
4EFBOFF,ACBOFF,AWBOFF,CBBOFF,MBDOFF,OCBOFF,CTBOFF,NWDBL
C
INTEGER DVPOFF,ARP0FF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
1SPDOFF,QVDOFF,CROFF,STDOFF,TRDOFF,QTOFF,QXTOFF,CTOFF,QTOFF,
2QTOFF,CTOFF,ACTOFF,RTOFF,PVTOFF,HFTOFF,BLDOFF,
3EFBOFF,ACBOFF,AWBOFF,CBBOFF,MBDOFF,OCBOFF,CTBOFF,QBOFF,QNOFF
C
COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
INTEGER TGP,BOT,RDBOT
DIMENSION ICDAT(11000)
EQUALNESS(ICDAT,CDAT)
C
CDAT(LBLK+QNOFF)=CDAT(LBLK+QNOFF)
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
ISTART=ICDAT(LBLKTB(1)+IBT-1)
IEND=ICDAT(LBLKTB(2)+IBT-1)
ACT=CDAT(LBLK+ACBOFF)+1
CDAT(LBLK+ACBOFF)=ACT
AWL=CDAT(LBLK+AWBOFF)
EF=ACT*(1.-((B1*AWL/ACT)+B2))
CDAT(LBLK+EFBOFF)=EF
ACT=ACT+1.
EF=ACT*(1.-((B1*AWL/ACT)+B2))
CDAT(LBLK+QNOFF)=OBJFUN(LPARM,ISTART, IEND, LPCT, LDAY, LTOUR, EF)
RETURN
END
Subroutine STRDF

Subroutine STRDF determines the change in objective function value per car-hour that would be realized by making an incremental change in the assignment of cars to a shift. LDAY and LTOUR are pointers to the day and shift. ITYPE is the relative position of the tour among all tours in the data base.

For any shift, a difference is computed by summing the contribution to the objective function of its blocks with the current number of cars and with one additional car assigned, and dividing by the length of the tour. For overlay shifts an additional difference is obtained, summing the current and proposed objective function values of the first block of the first overlaid shift and the second block of the second overlaid shift, subtracting the sums, and dividing by the difference between the sum of the lengths of the overlaid tours and the length of the overlay tour.
SUBROUTINE STRDF(LDAY, LTOUR, ITYPE)

C SUBROUTINE TO DETERMINE THE EFFECT ON THE OBJECTIVE FUNCTION OF
C ADDING A CAR TO A TOUR OR TAKING A CAR AWAY FROM AN OVERLAY TOUR
C AND ADDING A CAR TO EACH OF THE OVERLAID TOURS.

COMMON/PNTRS, IOVRLY, IOVTR(2),
  1NPCTOT, NPPCTRD, LPCTDT, LNMLSL(4), NNAMES(4), NDAYOT, LDAYNM,
  2LDRYFL, NDAYRD, LDWFL, NTRDF, LTRTB(2), LTRST, LTREND, LTRRFL, LTRNM,
  3NTRRO, LTRWFL, NBLDT, BLKTBI(2), BBLRF, NBLRD, LBFLW, NDIVOT, DIVRDI,
  4LDIVNM, LDIVFL

COMMON/OFFSET/NMPOFF, DVPOFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPOFF,
  1NWDPCT, CPDOFF, SPDFF, QVDFF, CRDOFF, SDTOFF, TRDOFF, NWDDY,
  2QDTOFF, QXTOFF, CRTOFF, QTOFF, QTDOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF,
  3PVTTOFF, HTTOFF, MFTOFF, LFTOFF, NPR10, NWDTR, BLDFF, QBOFF, QNBOFF,
  4EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, NWDBL

INTEGER DVPOFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPOFF, CPDOFF,
  1SPDFF, QVDFF, CRDOFF, SDTOFF, TRDOFF, QTOFF, QTDOFF, CRTOFF, QTOFF,
  2QDTOFF, CTTOFF, TYTOFF, ACTOFF, RVTOFF, PVTTOFF, HTTOFF, MFTOFF, LFTOFF,
  3EFBOFF, ACBOFF, AWBOFF, CRBOFF, RMBOFF, OCBOFF, CTBOFF, QBOFF, QNBOFF

COMMON/STOR/E/TOP, BOT, RDBOT, MAXBOT, NWDRS, CDAT(11000)
INTEGER TOP, BOT, RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT, CDAT)

ISTART=ICDAT(LTRST+ITYPE-1)
IEND=ICDAT(LTREND+ITYPE-1)
TOURLN=IEND-ISTART+1
CDAT(LTOUR+QTOFF)=(CDAT(LTOUR+QTOFF)-CDAT(LTOUR+QTOFF))/TOURLN
IF (ICDAT(LTOUR+TYTOFF) .NE. 5) RETURN
IF (ICDAT(LTOUR+ACTOFF) .GE. 1.) GO TO 10
CDAT(LTOUR+QTOFF)=-1.
RETURN

10 TOTLEN=0.
QOLD=0.*
QNEW=0.*
DO 20 I=1,2
ITRD=IOVTR(I)
ITP=ICDAT(LTRWFL+ITRRO-1)
IBDT=ICDAT(LTRTB(I)+ITP-1)
IBRD=ICDAT(LBLRF+IBDT-1)
LBKL=LDAY+BLDFF+(IBRD-1)*WDBL
QOLD=QOLD+CDAT(LBLK+QBOFF)
QNEW=QNEW+CDAT(LBLK+QNBOFF)
ISTART=ICDAT(LTRST+ITYPE-1)
IEND=ICDAT(LTREND+ITYPE-1)
TOTLEN=TOTLEN+(IEND-ISTART+1)

CONTINUE
CDAT(LTOUR+QTOFF)=(QOLD-QNEW)/(TOTLEN-TOURLN)
RETURN
END
Subroutine ADJUST

Subroutine ADJUST insures that the initial assignment of cars to shifts for an ALOC command results in the lowest possible objective function value. LPARM is a pointer to a parameter list that specifies the objective function. LPCT and LDAY are pointers to the data for the precinct and day. The absolute value of XDELT is the maximum number of cars that can be moved from an overlay shift to an overlaid shift to reduce the objective function value.

The sign of XDELT indicates whether cars can be moved to the first overlaid shift (positive) or the second overlaid shift (negative). No cars can be shifted if XDELT is zero or no cars are assigned to the overlay shift. Up to ABS(XDELT) cars are moved from the overlay shift to the appropriate overlaid shift. The process terminates when moving another car would increase the objective function value.
SUBROUTINE ADJUST(LPARM,LPCT,LDAY,XDELT)

C SUBROUTINE TO EXAMINE ALTERNATIVE INITIAL ALLOCATIONS FOR THE
C ALOC COMMAND TO FIND THE INITIAL ALLOCATION WITH THE BEST
C OBJECTIVE FUNCTION VALUE

C COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWJRS,CDAT(11000)
INTEGER TOF,BOT,RDBOT
DIMENSION ICDAT(11000)
EQUIVALENCE(ICDAT,CDAT)

C COMMON/OFFSET/NMONS,DVPOFF,APPOFF,SMPOFF,B1POFF,B2POFF,DYPF,  
1NWDCP,CPDOFF,SDOFF,DVDOFF,CRDOFF,STDFF,TRDOFF,NWDDY,
2QDTOFF,QXTOFF,CTTOFF,QTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RYTOFF,
3PVTOFF,HFTOFF,MFTOFF,LTOFF,FPRIOP,NWDTR,BLDOFF,QUOBOFF,GNBOFF,
4EFBOFF,ACBDOFF,AWBDOFF,CRBDOFF,RMOFF,OCBDOFF,CTBDOFF,NWDBL

C INTEGER DVPOFF,APPOFF,SMPOFF,B1POFF,B2POFF,DYPF,CPDOFF,
1SDOFF,DVDOFF,CRDOFF,STDFF,TRDOFF,QDTOFF,QXTOFF,CTTOFF,QTOFF,
2QDTOFF,CTTOFF,TYTOFF,ACTOFF,RYTOFF,MCFF,EFBOFF,CRBDOFF,
3RMOFF,OCBDOFF,CTBDOFF,QNBOFF

C COMMON/PNTR/IOVLREV,IOVTR(2),
1PNCTLD,PNCTRD,PNCTLD,PNLST,4,NAMES(4),NDAYDT,LDAYNM,
2LDIRFL,NDAYRD,LDIRFL,NDTRD,LTTRB(2),LTSTR,LTREND,LSMFR,LTRENM,
3NTRRD,LTSTRWFL,NLBDL,LBLKT(2),LBLRF,LNLRO,LBFLWL,NDIVD,NDIVRD,
4LDIVNM,LDIVFL

C FIND BLOCKS WHOSE ASSIGNMENTS CAN BE CHANGED

C LOVTR=LDAY+TRDOFF+(NTRRD-1)*NWDDY
ENOV=CDAT(LOVTR+ACTOFF)
IF(ENOV .LE. 0.) RETURN
ISW=1
IF(XDELT .LT. 0.) ISW=2
ITRRD=IOVTR(ISW)
ITYPE=ICDAT(LSTRWFL+ITRORD)
ILEN=ICDAT(LTRKEN+ITYPE-1)-ICDAT(LTRST+ITYPE-1)+1
IOVLEN=ICDAT(LTRND+LNDRT-1)-ICDAT(LTRST+LNDRT-1)+1
IF(ILEN .LT. IOVLEN) RETURN
DELTA=ABS(XDELT)
IF(DELTA .LT. 9999) RETURN
INV=2/ISW
IBOT2=ICDAT(LTRTB(INV)+NTRRT-1)
IBRD=ICDAT(LBLRF+IBOT2-1)
IBLK2=LDAY+BLDOFF*(IBRD-1)*NWDBL
IEND=ICDAT(LBLKTB(I)+IBOT2-1)
IBOT=ICDAT(LBLKTB(I)+IBOT2-1)
IBOT2+2*(-1)**ISW
IBRD=ICDAT(LBLRF+IBOT2-1)
IBLK1=LDAY+BLDOFF*(IBRD-1)*NWDBL
ILTOUR=LDAY+TRDOFF*(ITRORD-1)*NWDDY
ITRRD=IOVTR(INV)
LTTOR=LDAY+TRDOFF*(ITRORD-1)*NWDDY
ITYPE=ICDAT(LSTRWFL+ITRORD-1)
B1=CDAT(LPCT+B1POFF)
B2=CDAT(LPCT+B2POFF)
AWL = CDAT(LBLK2 + AWBOFF)

C ADJUST BLOCK AND TOUR ASSIGNMENTS TO MINIMIZE OBJECTIVE FUNCTION VALUE

C

10 QOLD = CDAT(LBLK1 + QBOFF) + CDAT(LBLK2 + QBOFF)
ACT = CDAT(LBLK2 + ACBOFF) - 1.
EF = ACT * (1. - ((B1 * AWL / ACT) + B2))
QTEST = OBKFUN(LPAM1, ISTART, IEND, LPCT, LDAY, LTTOUR, EF)
QNEW = CDAT(LBLK1 + QNBOFF) + QTEST
IF(QOLD .LT. QNEW) GO TO 100
CALL SBLOBJ(LPAM1, LPCT, LDAY, LTTOUR, LBLK1, IBDTI)
CDAT(LTTOUR + ACTOFF) = CDAT(LTTOUR + ACTOFF) + 1.
CDAT(LOVTR + ACTOFF) = CDAT(LOVTR + ACTOFF) - 1.
CDAT(LBLK2 + QNBOFF) = CDAT(LBLK2 + QBOFF)
CDAT(LBLK2 + ACBOFF) = ACT
CDAT(LBLK2 + EFBOFF) = EF
CDAT(LBLK2 + QBOFF) = QTEST
DELTA = DELTA - 1.
IF(DELTA .GT. 0.) GO TO 10
CALL STROBJ(LDAY, LOVTR, NTRDT)
CALL STROBJ(LDAY, LTTOUR, ITYPE)
RETURN
END
Subroutine STROBJ

Subroutine STROBJ determines the contribution of one shift to the objective function value and the difference in its contribution per car-hour if an additional car were assigned to the shift. LDAY and LTOUR are pointers to the data for the day and shift. ITYPE is the tour to which the shift belongs.

The objective function contributions of the shift are determined by summing the contributions of its blocks. Subroutine STRDF is called to determine the improvement per car-hour that would be realized if one car were added to the shift. (STRDF also determines the improvement per car-hour that would be realized by removing a car from an overlay shift and adding one car to each of the shifts that it overlays.)
SUBROUTINE STROBJ(LDAY,LYOUR,ITYPE)
C
C EVALUATES A WEIGHTED OBJECTIVE FUNCTION FOR ONE SHIFT.
C
COMMON/STORE/TOP,BOT,ROBOT,MAXBOT,NWDRS,CDAT(11000)
INTEGER TOP,BOT,ROBOT
DIMENSION ICDAT(11000)
EQUIVALENCE (ICDAT,CDAT)
C
COMMON/PNTRS/IOVRLY,IOVTR(2),
1NPCTDT,NPCTR,LPCTDT,LNMLST(4),NNAMES(N),NDAYDT,LDAYNM,
2LDYRF,NDAYRD,LDYRF,NTRTD,LTRTB(2),LTRST,LTREN,LTREF,LTRNM,
3NTRD,LTRWF,NBLDT,LBLKTB(2),LBLRF,NBLRD,LBLRF,NDIVD,NDIVR,
4LDIVNM,LDIVFL
C
COMMON/OFFSET/NMPOFF,DVPFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
2QTOFF,QTFF,CTTOFF,QTFF,CTTOFF,QTFF,CTTOFF,QTFF,CTTOFF,QTFF,
3PVTFF,HHTFF,MTTOFF,LHTFF,NPRIO,NWDT,BLDOFF,GOBFF,QNBOFF,
4EBOFF,ACBOFF,AWBOFF,CRBOFF,MBBOFF,OCBOFF,CTBOFF,NWDBL
C
INTEGER DVPFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPFF,CPDOFF,
1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QTOFF,QTFF,CTTOFF,QTFF,
2QTOFF,CTTOFF,QTFF,CTTOFF,QTFF,CTTOFF,QTFF,CTTOFF,QTFF,
3EBOFF,ACBOFF,AWBOFF,CRBOFF,MBBOFF,OCBOFF,CTBOFF,GOBFF,QNBOFF
C
QOLO=0.
QNEW=0.
DO 10 IBLK=1,2
IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)
IF(IBDT .LT. 1) GO TO 10
IBRD=ICDAT(LBLRF,IBDT-1)
LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
QOLO=QOLO+CDAT(LBLK+QOBOFF)
QNEW=QNEW+CDAT(LBLK+QNBFF)
10 CONTINUE
CDAT(LTOUR+QTOFF)=QOLO
CDAT(LTOUR+QTOFF)=QNEW
CALL STRDF(LDAY,LTOUR,ITYPE)
RETURN
END
Function OBJFUN

Function OBJFUN (objective function) evaluates an objective function over a span of hours of a day. Parameter LPARM is a pointer to a number list that specifies the objective function to be evaluated and any associated parameters. ISTART and IEND specify the span of hours over which the function is to be evaluated. LPCT, LDAY, and LTOUR are pointers to the precinct, day, and tour in which the span of hours occurs. EF is the number of effective cars for which the function is to be evaluated.

OBJFUN selects the correct function subprogram to evaluate the objective function specified by LPARM. For objective function 2, the second element of LPARM is the priority for which it is to be evaluated.
FUNCTION OBJFUN(LPARAM, ISTART, IEND, LPCT, LDAY, LTOUR, EF)  3439
C  3440
C EVALUATES AN OBJECTIVE FUNCTION OVER THE SPAN OF HOURS  3441
C FROM ISTART TO IEND  3442
C  3443
COMMON/STORE/TOP, BOT, RDBOT, MAXBOT, NWORDS, CDAT(11000)  3444
INTEGER TOP, BOT, RDBOT  3445
DIMENSION IDCAT(11000)  3446
EQUIVALENCE (IDCAT, CDAT)  3447
C  3448
COMMON/OFFSET/NUMOFF, DVPOFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPOFF,  3449
1NWPCT, CPDOFF, SPD0FF, OVD0FF, CRDOFF, SDDOFF, TRDOFF, NWDOY,  3450
2QDTOFF, QXTOFF, CRTOFF, QTOFF, QNTOFF, CTTOFF, YTTOFF, ACTOFF, RVTOFF,  3451
3PVTOFF, HFTOFF, MFTOFF, LFTOFF, NPRIO, NWDTR, BLDOFF, QBOFF, QNBOFF,  3452
4EFBOFF, ACBOFF, ABBOFF, CRBOFF, RBBOFF, QCB0FF, CTBOFF, NWDBL  3453
C  3454
INTEGER DVPOFF, ARPOFF, SMPOFF, B1POFF, B2POFF, DYPOFF, CPDOFF,  3455
1NWPCT, CPDOFF, SPD0FF, CRDOFF, SDDOFF, TRDOFF, QDTOFF, QXTOFF, CRTOFF,  3456
QTOFF, QNTOFF, CTTOFF, YTTOFF, ACTOFF, RVTOFF, PVTOFF, HFTOFF, BLDOFF,  3457
3EFBOFF, ACBOFF, ABBOFF, CRBOFF, RBBOFF, QCB0FF, CTBOFF, QBOFF, QNBOFF  3458
C  3459
COMMON/PNTRS/IVRRLY, IVT(2),  3460
1NPCTDT, NPTCDT, LNPCTDT, LNMLST(4), N NAMES(4), NDYD, LDAYNM,  3461
2LDYRL, NDAYRD, LDYWFL, NTRD, LTRTB(2), LTRST, LTREND, LTRRFL, LTRNM,  3462
3NTRRD, LTRWFL, NBNDT, LBLKT(2), LBLRFL, NBLRD, LBLWFL, NDIVDT, NDIVRD,  3463
4LDIVNM, LDIVFVL  3464
C  3465
IFNCTN=IDCAT(LPARAM)  3466
LCR=LDAY+CRDOFF  3467
LST=LDAY+STDOFF  3468
LFR=LTOUR+HTOFF  3469
GO TO (10, 20, 30), IFNCTN  3470
C  3471
10 OBJFUN=OBJF1(ISTART, IEND, LCR, LST, EF)  3472
GO TO 100  3473
20 N=IDCAT(LPARAM+2)  3474
OBJFUN=OBJF2(N, ISTART, IEND, LCR, LST, LFR, EF)  3475
GO TO 100  3476
30 RV=CDAT(LTOUR+RVTOFF)  3477
OBJFUN=OBJF3(ISTART, IEND, LPCT, LDAY, RV, EF)  3478
100 RETURN  3479
END  3480
Subroutine ADDCAR

Subroutine ADDCAR (add cars) adds cars to a set of shifts so that the average value of a specified objective function is minimized. Parameter LPARAM is a pointer to a number list that specifies the function to be evaluated. NCARHR is the number of car-hours available for allocation.

The allocation algorithm used is described in Appendix B of the User's Manual.
SUBROUTINE ADDCAR(NCARHR,LPARM)

C ADDS CARS TO A SET OF SHIFTS SO THAT THE AVERAGE VALUE
C OF A SPECIFIED OBJECTIVE FUNCTION IS MINIMIZED

C
C COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWRODS,CDAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICAT(11000)
EQUIVALENCE (ICAT,CDAT)
C
C COMMON/PNTRS/IOVRLY,IOVTR(2),
1NPCTDT,NPCTRDL,PCTDT,LPNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRF,LNDAYRD,LDYWF,LTRDT,LTRTB(2),LFRST,LTREND,LTRRFL,LTRNM,
3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRF,LNBLRD,LBLWF,LNDVDT,LNDVRO,
4LDIVNM,LDIVFL
C
COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPPOFF,B1POFF,B2POFF,DYPOFF,
1NWPCT,CPDOFF,SPDOFF,QVDOFF,CRDOFF,STDROF,TRDOFF,NDDYY,
2QTOFF,QTOFF,CTOFF,QTQTOFF,QNTOFF,CXTOFF,CTOFF,CTOFF,ACTOFF,RTQTOFF,
3FTOFF,FTOFF,MTOFF,LTFF,LPROD,TDD,LCOFF,QBOFF,QNOFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMOBOF,OCBOFF,CBOFF,NWDBL
C
INTEGER DVPFF,ARPOFF,SMPPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
1SPDOFF,QVDOFF,CRDOFF,STDROF,TRDOFF,NDDYY,
2QTOFF,CTOFF,QTOFF,CTOFF,CTOFF,CTOFF,CTOFF,CTOFF,ACTOFF,RTQTOFF,
3FTOFF,FTOFF,MTOFF,FTOFF,FTOFF,FTOFF,FTOFF,FTOFF,FTOFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMOBOF,OCBOFF,CBOFF,CTBOFF,CTBOFF,QBOFF
C
C COMMON/SYSTEM/SYSIN,SYSOUT,FILE,LIT
INTEGER SYSIN,SYSOUT
C
NLEFT=NCARHR
C
C FIND SHIFT WITH GREATEST IMPROVEMENT PER CAR HOUR IN
C OBJECTIVE FUNCTION VALUE IF ALLOCATION IS CHANGED
C INCREMENTALLY.
C
310  LBPTC=NXPCT(0)
LBDAY=NXDAY(LBPTC,0)
LBTOUR=NXTOUR(LBDAY,0,IBTYPE)
QBIG=CDAT(LBTOUR+QTOFF)
LPCT=0

320  LPCT=NXPCT(LPCT)
IF(LPCT EQ 0) GO TO 350
LDAY=0

330  LDAY=NXDAY(LPCT,LDAY)
IF(LDAY EQ 0) GO TO 320
LTTOUR=0

340  LTOUR=NXTOUR(LDAY,LTTOUR,ITYPE)
IF(LTOUR EQ 0) GO TO 330
QDIFF=AMAX1(CDAT(LTOUR+QTOFF),CDAT(LTOUR+QTOFF))
IF(QDIFF LE QBIG) GO TO 340
QBIG=QDIFF
IBTYPE=ITYPE
LBTOUR=LTOUR
LBDAY=LDAY
LBPTC=LPCT
GO TO 340
C
350  IF(LBTOUR .NE. 0) GO TO 360
      WRITE(SYSOUT,5)
      FORMAT('/** *** NO SHIFTS SELECTED - REENTER**/
      RETURN
360  IF(IDCAT(LBTOUR+TYTOFF) .EQ. 5 .AND. CDAT(LBTOUR+QXTOFF)
      .GT. CDAT(LBTOUR+QOTFF)) GO TO 500
      ILEN=IDCAT(LTREND+IBTYPE-1)-IDCAT(LTSTR+IBTYPE-1)+1
      IF(ILEN .GT. NLEFT) RETURN
C
C ADD A CAR TO SELECTED SHIFT AND COMPUTE NEW OBJECTIVE
      FUNCTION VALUE
C
      CDAT(LBTOUR+ACTOFF)=CDAT(LBTOUR+ACTOFF)+1.
      NLEFT=NLEFT-ILEN
      CDAT(LBTOUR+QOTFF)=CDAT(LBTOUR+QOTFF)
      CDAT(LBTOUR+QXTOFF)=0.
      DO 370 IB=1,2
         IBOT=IDCAT(LTRT8(IB)+IBTYPE-1)
         IF(IBOT .LT. 1) GO TO 370
         IBRD=IDCAT(LBLRFL+IBDT-1)
         LBLK=LBDAY+BLDOFF+(IBRD-1)*NWDBL
         LTTOUR=LBTOUR
         IF(IDCAT(LBTOUR+TYTOFF) .EQ. 5)
      1     LTTOUR=LBDAY+TRDOFF+(IDVTR(IB)-1)*NWDT
      CALL SBLOBJ(LPARAM,LBPCT,LBDAY,LTTOUR,LBLK,IBDT)
      CDAT(LBTOUR+QXTOFF)=CDAT(LBTOUR+QXTOFF)+CDAT(LBLK+QNOFF)
      CONTINUE
      CALL STRDF(LBDAY,LBTOUR,IBTYPE)
      ID=IDCAT(LBTOUR+TYTOFF)-1
      GO TO (310,390,390,410),10
C
C ADJUST OBJECTIVE FUNCTION DIFFERENCES FOR SHIFTS IN
      OVERLAY SEGMENTS
C
390  ITRRD=IDCAT(LTRRFL+NTROT-1)
      LTTOUR=LBDAY+TRDOFF+(ITRRD-1)*NWDT
      CALL STROBJ(LBDAY,LTTOUR,NTROT)
      GO TO 310
C
410  DO 420 I=1,2
      ITRRD=IDVTR(I)
      ITYPE=IDCAT(LTRWFL+ITRD-1)
      LTTOUR=LBDAY+TRDOFF+(ITRD-1)*NWDT
      CALL STROBJ(LBDAY,LTTOUR,ITYPE)
      GO TO 310
C
C DECREASE OVERLAY SHIFT ASSIGNMENT AND INCREASE ASSIGNMENTS
      TO OVERLAID SHIFTS
C
500  ITOT=0
      DO 510 I=1,2
      ITRRD=IDVTR(I)
      ITP=IDCAT(LTRWFL+ITRRD-1)
      ISTART=IDCAT(LTSTR+ITP-1)
      IEND=IDCAT(LTREND+ITP-1)
      ITOT=ITOT+IEND-ISTART+1
      ILEN=ITOT-(IDCAT(LTREND+IBTYPE-1)-IDCAT(LTSTR+IBTYPE-1)+1)
      GO TO 510
IF(ILEN .GT. NLEFT) RETURN
NLEFT=NLEFT-ILEN
CDAT(LBTOR+ACTOFF)=CDAT(LBTOR+ACTOFF)-1.
DO 520 I=1,2
ITRDD=IOVT(I)
ITP=ICDAT(LTRWF+ITRDD-1)
IBDT=ICDAT(LTRTB(I)+ITP-1)
IBRD=ICDAT(LBLRFL+IBDT-1)
LBLK=LBDAY+BLDOFF+(IBRD-1)*NWDBL
LTTOR=LBDAY+TRDOFF+(ITRDD-1)*NWDR
CDAT(LTTOR+ACTOFF)=CDAT(LTTOR+ACTOFF)+1.
CALL SBLOBJ(LPARM,L8PCT,LBDAY,LTTOR,LBLK,IBDT)
CALL STROBJ(LBDAY,LTTOR,ITP)
CONTINUE
CALL STRDF(LBDAY,LTTOR,IBTYPE)
GO TO 310
END
Subroutine WRITE

Subroutine WRITE implements the WRITE command. It writes a file on a user-specified unit number which can later be used as a DATABASE file. Data are written only for precincts, days, and tours specified in the command qualifier.
SUBROUTINE WRITE

C SUBROUTINE IMPLEMENTS WRITE COMMAND
C
COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
INTEGER TYPOFF,WDTYPE
DIMENSION PCLSNM(8),OCLSNM(8),TOURNM(8)
EQUIVALENCE (PCLSNM,KEYWD(1,4)),(OCLSNM,KEYWD(1,3)),
(TOURNM,KEYWD(1,2))

COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,COAT(11000)
INTEGER TOP,BOT,RDBOT
DIMENSION ICOAT(11000)
EQUIVALENCE (ICOAT,COAT)

COMMON/OFFSET/NMPOFF,DVPOFF,ARPPOFF,MPPOFF,B1POFF,B2POFF,DYPOFF,
INWDPCT,CPDOFF,SPDOFF,QVDOFF,CRDOFF,STDFF,TRDOFF,NWDDY,
QDTOFF,QXTOFF,QCTOFF,QTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDIR,BLDOFF,QLBOFF,QNBOFF,
+EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

INTEGER DVPOFF,ARPPOFF,MPPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
INWDPCT,CPDOFF,SPDOFF,QVDOFF,CRDOFF,STDFF,TRDOFF,QDTOFF,
QXTOFF,QCTOFF,QTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDIR,BLDOFF,QLBOFF,QNBOFF,
+EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QBOFF,QNBOFF

COMMON/PNTRS/IOVRLY,IOVTR(2),
INPCTOT,PNCTRD,LPCDTOT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRFL,NDAYRD,LDYWFL,NTROT,LTRTB(2),LTRSL,LTRRFL,LTRNM,
3NTRD,LTRWFL,NBLDT,LBLKTB(2),BLRLF,NSLRD,LBLWFL,NOIVDT,NOIVRD,
+LDIVN4,LDIVFL

COMMON/SYSTEM/SYSIN,SYSDOUT,IFILE,LIT
INTEGER SYSIN,SYSDOUT

COMMON/SCODES/SEND,CMO,NUMLST,NAMLST,FSPEC,DSPEC,DU,ERR
INTEGER SEND,CMO,FSPEC,DSPEC,DU,ERR

INTEGER TYPE,VAL
DIMENSION VAL(2),ORDER(3)
DATA BLANK/'1H /

LGETT=TOP
TYPE=CMD
CALL SCAN(TYPE,VAL)

GET UNIT NUMBER

IF(TYPE .EQ. NUMLST) GO TO 20
WRITE(SYSOUT,9010)
9010 FORMAT('*** INVALID UNIT SPECIFICATION - REENTER.*')
TOP=LGETT
RETURN
20 NUNIT=ICOAT(VAL(2))
IF(NUNIT .EQ. SYSIN .OR. NUNIT .EQ. SYSDOUT .OR. NUNIT .EQ. IFILE)
1 OR. NUNIT .LT. 1 OR. NUNIT .GT. 99) GO TO 10
SCAN QUALIFIER

CALL SCAN(TYPE,VAL)
CALL GTOSEQ(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 30
TOP=LGETT
RETURN

SET WORK FLAGS

30 CALL SETWFL(IERR)
IF(IERR .EQ. 0) GO TO 35
TOP=LGETT
RETURN

CHECK OVERLAY SPECIFICATION

35 CALL CKOVR(IERR)
IF(IERR .EQ. 0) GO TO 40
TOP=LGETT
RETURN

DETERMINE NUMBER OF DAY, TOURS AND PRECINCTS
IN NEW DATA BASE

40 NDAY=NNAMES(1)
IF(NDAY .LT. 1) NDAY=NDAYRD
NTOUR=NNAMES(2)
IF(NTOUR .LT. 1) NTOUR=NTRRD
NPCT=0
LPCT=0
LPCT=NXPCT(LPCT)
IF(LPCT .LE. 0) GO TO 55
NPCT=NPCT+1
GO TO 50

CONTINUE

IOVT=10VRLY
IF(ICODEAT(LTRRFL+NTRDT-1) .EQ. 0 .OR. (ICODEAT(LTRRFL+NTRDT-1) .EQ. 1 .AND. ICODEAT(LTRRFL+NTRRD-1) .EQ. 0)) IOVT=0

WRITE CONTROL RECORD

WRITE(NUNIT,1) DCLSNM,PCLSNM,TOURNM,NDAYRD,NPCT,NDAY,NBLDT,
1 NTOUR,IOVT
CALL GETTOP(80,LREC)
I=0
K=LREC-1
DO 60 IDAY=1,NDAYRD
ID=ICODEAT(LDYWFL+IDAY-1)
IF(ID .LT. 1) GO TO 60
LNW=(ID-1)*8+LDAYNM
CALL MVE(I,ICODEAT(LNM),ICODEAT(LREC+1),8)
I=I+8
IF(I .LT. 80) GO TO 60

WRITE DAY NAMES

WRITE(NUNIT,2)(ICODEAT(K+J),J=1,80)
I=0
CONTINUE
IF(I .GE. 0) WRITE(NUNIT,2) (ICDAT(K+J),J=1,I)
K=LBLKTB(2)-1
C
WRITE BLOCK DESCRIPTOR RECORDS
C
WRITE(NUNIT,3) (ICDAT(K+I),I=1,NBLDT)
DO 70 ITQR=1,NTRRD
IT=ICDAT(LTRWF+ITQR-1)
IF(IT .LT. 1) GO TO 70
IT=IT-1
LNM=IT*8+LTRNM-1
C
WRITE TOUR DESCRIPTOR RECORDS
C
WRITE(NUNIT,4) (ICDAT(LNM+I),I=1,8),ICDAT(LTRTB(1)+IT),
1 ICDAT(LTRTB(2)+IT)
70 CONTINUE
C
LPCT=0
100 LPCT=NXPCT(LPCT)
IF(LPCT .NE. 0) GO TO 110
ENDFILE NUNIT
TOP=LGETT
RETURN
110 IDIV=ICDAT(LPCT+DVOFF)-1
LPCTNM=LPCT+NMPFF-1
LNVNM=LDIVNM+IDIV*8-1
C
WRITE PRECINCT HEADER
C
WRITE(NUNIT,5) (ICDAT(LPCTNM+I),I=1,8),(ICDAT(LNVNM+I),I=1,8),
1 (CDAT(LPCT+I),I=ARPFF,82POFF)
C
LDAY=0
200 LDAY=NXDAY(LPCT,LDAY)
IF(LDAY .LT. 1) GO TO 100
CPARM=CDAT(LDAY+CPDOFF)
SPARM=CDAT(LDAY+SPDOFF)
C
WRITE DAY DETAIL RECORDS
C
WRITE(NUNIT,6) CPARM,SPARM,ICDAT(LDAY+QVDOFF)
LCRE=LREC-1
LSTO=LREC+23
LCRI=LDAY+CROFF-1
LSTI=LDAY+STDFF-1
SPARM=SPARM/60.
DO 210 I=1,24
ICDAT(LCRO+I)=100.*(0.005+CDAT(LCRI+I)/CPARM)
210 ICDAT(LSTO+I)=100.*(0.005+CDAT(LSTI+I)/SPARM)
WRITE(NUNIT,7) (ICDAT(LCRO+I),I=1,48)
DO 220 I=1,24
220 ICDAT(LCRO+I)=0
C
LTOUR=0
300 LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
```fortran
IF(LTOUR .NE. 0) GO TO 320
IF(IOVT .NE. 0 .AND. ICDAT(LDAY+QVDJFF) .EQ. 0)
   1 WRITE(NUNIT,2) BLANK
      DO 310 I=1,NBLDT
         IBLK=ICDAT(LBLRFL+I-1)
         IF(IBLK .LE. 0) GO TO 310
         LBLK=LDAY+BLDFF+(IBLK-1)*WDBL
         CDAT(LCRO+I)=CDAT(LBLK+QCBJFF)
      CONTINUE
      310 WRITE BLOCK DETAIL RECORD
      WRITE(NUNIT,8) (CDAT(LCRO+I),I=1,NBLDT)
      GO TO 200
   C WRITE SHIFT DETAIL RECORD
   C 320 WRITE(NUNIT,9) (CDAT(LTOUR+I),I=ACTOFF,MFTOFF)
      GO TO 300
      1 FORMAT(2(8A1,2X),8A1,1X,I2,1X,I3,1X,I3,1X,1,2(I2,1X),11)
      2 FORMAT(8A1)
      3 FORMAT(24(I2,1X))
      4 FORMAT(8A1,1X,I2,1X,I2)
      5 FORMAT(8A1,1X,8A1,2X,F5.2,1X,F5.1,2(I1,F5.3))
      6 FORMAT(2(F5.2,1X),11)
      7 FORMAT(24I3)
      8 FORMAT(24F3.1)
      9 FORMAT(3(F5.1,1X),2(F5.4,1X))
      END
```
Subroutine SKIP

Subroutine SKIP is called to skip past N records in the data file that are not needed by the calling routine.

```
SUBROUTINE SKIP(IFILE,N)
  C     SKIPS N RECORDS IN FILE IFILE
  C
  IF( N.LT.1) RETURN
  DO 1 I=1,N
  1 READ(IFILE,2) J
  2 FORMAT(A1)
  RETURN
END
```
Function CEIL

CEIL(X) is the least integer that is greater than or equal to X.

FUNCTION CEIL(X)
C
C LEAST INTEGER GREATER THAN OR EQUAL TO X
C
ICEIL=X
CEIL=ICEIL
IF(X.GT.ICEIL)CEIL=CEIL+1.
RETURN
END
BLOCK DATA

BLOCK DATA establishes the initial numerical values of variables in COMMON blocks used by the program's subroutines and functions.

The COMMON blocks are as follows:

- COMMON/PNTRS/
  - Pointers. See Chapter IV.
- COMMON/OFFSET/
  - Offsets. See Chapter IV.
- COMMON/STORE/
  - Parameters related to run-time storage requirements.
- COMMON/SYSTEM/
  - Input and output unit numbers.
- COMMON/KEYWDS/
  - Keywords and word types.
- COMMON/LCODES/
  - Lexical types returned by GETTKN.
- COMMON/SCODES/
  - Syntactic types returned by SCAN.
- COMMON/STATS/
  - Statistics and output orders.
BLOCK DATA

C
COMMON/PNTR,IOVRLY,IOVTR(2),
1NPCTOT,NPCTR,D,LCTOT,LMNLST(4),NNAMES(4),NDAYDT,LDAYNM,
2LDYRF,LNDAYRD,LNDFUL,NDAR,LTRRB(2),LTRST,LTREND,LTRFFL,LTRNM,
3NTRD,LTRFNL,NBLOD,LBLKTB(2),LBLRF,NBLRD,LBNLFL,NDIVDT,NDIVRD,
4LDIVNM,LDIVFL
DATA NDAYRD/0/,NTRRD/0/

C
COMMON/OFFSET/NMPOFF,DVPOFF,ARPPOFF,SMPOFF,81POFF,B2POFF,DYPOFF,
1NWDPCT,CPDOFF,SPDOFF,OVDDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
2QTOFF,QXTOFF,CRTOFF,QTDOFF,QTDOFF,CRTOFF,QTDOFF,CTTGF,
3PVTOFF,HFTOFF,MTDFF,LFTOFF,NPRIO,NDTR,BLDOFF,QBOFF,QNOFF,
4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBF,CTBOFF,NWDBL

C
INTEGER DVPOFF,ARPPOFF,SMPOFF,81POFF,B2POFF,DYPOFF,CPDOFF,
1SPDOFF,OVDDOFF,CRDOFF,STDOFF,TRDOFF,QTDOFF,CRTOFF,QTDOFF,
2QTOFF,CRTOFF,QTDOFF,QTDOFF,PVTOFF,QTOFF,MTDFF,LTFF,
3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBF,CTBOFF,QBOFF,QNOFF

C
DATA NMPOFF/0/,DVPOFF/8/,ARPPOFF/9/,SMPOFF/10/,81POFF/11/,B2POFF/12/,DVPOFF/13/,CPDOFF/0/,SPDOFF/1/,OVDFOFF/2/,CRDOFF/3/,STDOFF/27/,TRDOFF/51/,QTDOFF/0/,QXTDOFF/1/,CRTOFF/2/,QTOFF/3/,QTOFF/4/,QTOFF/5/,QTOFF/6/,QTOFF/7/,QTOFF/8/,PVTDOFF/9/,HFTDOFF/10/,MTDOFF/11/,LFTDOFF/12/,NPRIO/3/,NDTR/13/,EFBOFF/0/,ACBOFF/1/,AWBOFF/2/,CRBOFF/3/,RMBOFF/4/,OCBF/5/,CTBOFF/6/,QBOFF/7/,QNOFF/8/,NWDBL/9/

C
COMMON/STOR,STOR,TOP,BOT,RDBOT,MAXB,WORD,MAXB,CDAT/01000/0
INTEGER TOP,BOT,RDBOT
DIMENSION ICDAT/10000/
EQUIVALENCE (ICDAT,CDAT)
DATA BOT/1/,NWORDS/100000/,MAXB/0/

C
COMMON/SYSTEM/SYLSIN,SYSCUT,ILINE,LIKE
INTEGER SYLSIN,SYSCOUT
DATA SYLSIN/4/,SYSCOUT/5/,ILINE/19/,LIKE/20/

C
COMMON/KEYWDS/NKWD,NNTYPES,TPPOFF(4),KEYWD(8),WDTYPE(30)
INTEGER TPPOFF,WDTYPE
DIMENSION PCLSNM(8),DCLSNM(8),TOURN(8)
EQUIVALENCE (PCLSNM,KEYWDS(1,4)),(DCLSNM,KEYWDS(1,3))
DATA TOURN(1),KEYWD(1,2)/
DATA NTYPES/4/,NKWD/26/,TPPOFF(1)/4/,TPPOFF(2)/4/,TPPOFF(3)/9/,TPPOFF(4)/9/

C
COMMON/LCODES/LEND,WORD,NUM,LP,RP
INTEGER WORD,RP
DATA LEND/1/,WORD/2/,NUM/3/,LP/4/,RP/5/

C
COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR
DATA SEND/5/,CMD/3/,NUMLST/6/,NAMLST/7/,FSPEC/2/,DSPEC/1/,DUM/4/,ERR/8/
1DUM/4/,ERR/8/
COMMON/STATS/T(4, 8), S(4, 8), PORDER(3), RORDER(3), CIND(8)
INTEGER PORDER, RORDER
DATA PORDER(1)/3/, PORDER(2)/2/, PORDER(3)/1/
END
Appendix A

DEMONSTRATION DATA BASE
DIVISION  PRECINCT  TOUR  02 005 007 05 04 1
SUN-MON  MON-TUE  TUE-WED  WED-THU  THU-FRI  FRI-SAT  SAT-SUN
08 11 16 19 24
MIDDAY  01
PM  02 03
AM  04 05
FOURTH  03 04
EAST  HIGHLAND  34.88  503.4  45.84  503.4
7.71  40.23  1
22.74  81.11  1.17  81.11  1.17  01.41  01.41  01.41  01.41  01.41  01.41  01.41  01.41
91.91  91.91  91.91  91.91  81.81  81.81  81.81  81.81  81.81  81.81  81.81  81.81  81.81  81.81
9.9  15.0  7.5  .065  .84
9.9  15.0  7.5  .065  .84
8.8  25.0  7.5  .065  .84
4.4  15.0  7.5  .065  .84
4.52  57.01  51.3
7.71  45.08  1
55.91  71.71  31.11  01.2  88.13  21.71  41.11  11.23  31.71  68.45  36.19  23
1.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01
12.9  15.0  7.5  .065  .84
9.9  15.0  7.5  .065  .84
9.9  25.0  7.5  .065  .84
3.8  15.0  7.5  .065  .84
4.02  58.21  0
7.71  43.91  1
88.62  65.84  1.19  71.91  11.71  31.41  31.79  1.04  81.45  52.39  29.19  49
1.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11
13.4  15.0  7.5  .065  .84
13.4  15.0  7.5  .065  .84
9.5  25.0  7.5  .065  .84
5.4  15.0  7.5  .065  .84
3.62  84.02  0.13
7.71  45.62  1
1.29  58.78  38.3  681.0  94.1  21.31  31.61  11.2  97.55  65.42  23.08  23.36
1.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01
14.2  15.0  7.5  .065  .84
11.6  15.0  7.5  .065  .84
9.8  25.0  7.5  .065  .84
4.1  15.0  7.5  .065  .84
4.03  83.31  5.50
7.71  41.19  1
1.1  97.97  97.97  91.11  31.91  31.91  31.91  31.91  31.91  31.91  31.91  31.91  31.91  31.91
1.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11  11.11
12.9  15.0  7.5  .065  .84
11.6  15.0  7.5  .065  .84
9.8  25.0  7.5  .065  .84
5.4  15.0  7.5  .065  .84
4.02  53.52  5.50
7.71  43.92  1
71.58  65.71  75.78  1.01  29.11  71.21  51.42  12.21  29.17  75.75  49.32  24.36  42
97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97  97.97
12.1  15.0  7.5  .065  .84
11.6  15.0  7.5  .065  .84
10.6  25.0  7.5  .065  .84
5.5  15.0  7.5  .065  .84
5.72  37.02  0.50
7.71  37.91  1
1.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01  01.01
11.8  15.0  7.5  .065  .84
(continued)
<table>
<thead>
<tr>
<th>Value</th>
<th>10.6</th>
<th>15.0</th>
<th>7.5</th>
<th>0.65</th>
<th>0.84</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8.8</td>
<td>25.0</td>
<td>7.5</td>
<td>0.65</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>5.8</td>
<td>15.0</td>
<td>7.5</td>
<td>0.65</td>
<td>0.84</td>
</tr>
<tr>
<td>4.05</td>
<td>3.30</td>
<td>0.1</td>
<td>81.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WFST</td>
<td>LOWLAND</td>
<td>51.92</td>
<td>678.5</td>
<td>-0.476</td>
<td>0.637</td>
</tr>
<tr>
<td>6.65</td>
<td>37.88</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>38.60</td>
<td>94.53</td>
<td>831.11</td>
<td>31.31</td>
<td>51.71</td>
</tr>
<tr>
<td></td>
<td>0.90</td>
<td>90.90</td>
<td>90.90</td>
<td>90.90</td>
<td>90.90</td>
</tr>
<tr>
<td>10.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>9.6</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>8.6</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5.33</td>
<td>32.32</td>
<td>51.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>39.51</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>98.94</td>
<td>79.98</td>
<td>48.96</td>
<td>46.68</td>
<td>41.31</td>
</tr>
<tr>
<td>1.21</td>
<td>21.21</td>
<td>21.21</td>
<td>21.21</td>
<td>21.21</td>
<td>20.90</td>
</tr>
<tr>
<td>11.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>11.6</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>1.86</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>4.22</td>
<td>33.33</td>
<td>50.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>45.13</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>711.26</td>
<td>68.49</td>
<td>54.14</td>
<td>41.11</td>
<td>51.21</td>
</tr>
<tr>
<td></td>
<td>0.80</td>
<td>80.80</td>
<td>80.80</td>
<td>80.80</td>
<td>80.81</td>
</tr>
<tr>
<td>10.7</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>10.3</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>9.1</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>4.52</td>
<td>33.01</td>
<td>32.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>37.41</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>53.68</td>
<td>68.64</td>
<td>98.91</td>
<td>21.0</td>
<td>79.41</td>
</tr>
<tr>
<td></td>
<td>0.90</td>
<td>90.90</td>
<td>90.90</td>
<td>90.90</td>
<td>90.90</td>
</tr>
<tr>
<td>10.8</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>10.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>8.8</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>31.82</td>
<td>20.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>38.01</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>41.11</td>
<td>0.05</td>
<td>61.29</td>
<td>90.98</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>1.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>12.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>11.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>3.02</td>
<td>0.03</td>
<td>80.80</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>42.21</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>711.19</td>
<td>64.64</td>
<td>51.17</td>
<td>77.71</td>
<td>931.41</td>
</tr>
<tr>
<td></td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>11.1</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5.9</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>7.03</td>
<td>0.02</td>
<td>30.80</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>36.18</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.183</td>
<td>1.06</td>
<td>791.47</td>
<td>971.26</td>
<td>71.12</td>
</tr>
<tr>
<td></td>
<td>1.21</td>
<td>21.21</td>
<td>21.21</td>
<td>21.21</td>
<td>20.90</td>
</tr>
<tr>
<td>9.2</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>10.5</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>25.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>5.7</td>
<td>15.0</td>
<td>7.5</td>
<td>0.61</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>2.52</td>
<td>34.32</td>
<td>31.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(continued)
<table>
<thead>
<tr>
<th></th>
<th>10.6</th>
<th>15.0</th>
<th>7.5</th>
<th>567</th>
<th>584</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>3.02</td>
<td>35.33</td>
<td>0.8</td>
<td>550</td>
<td>47.43</td>
<td>1</td>
</tr>
<tr>
<td>1.28</td>
<td>41.2</td>
<td>821.11</td>
<td>0.071</td>
<td>11.41</td>
<td>61.41</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>1.1.0</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>5.52</td>
<td>85.31</td>
<td>0.8</td>
<td>550</td>
<td>47.43</td>
<td>1</td>
</tr>
<tr>
<td>5.50</td>
<td>45.71</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.97</td>
<td>86.91</td>
<td>91.01</td>
<td>96.11</td>
<td>61.61</td>
<td>61.91</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>1.26</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>6.9</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>6.8</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>4.02</td>
<td>30.01</td>
<td>0.3</td>
<td>5.50</td>
<td>44.77</td>
<td>1</td>
</tr>
<tr>
<td>8.66</td>
<td>96.50</td>
<td>36.50</td>
<td>45.82</td>
<td>11.21</td>
<td>71.51</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>11.8</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>2.52</td>
<td>20.0300</td>
<td>0.8</td>
<td>550</td>
<td>47.43</td>
<td>1</td>
</tr>
<tr>
<td>5.50</td>
<td>41.55</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>91.1</td>
<td>77.2</td>
<td>73.96</td>
<td>961.31</td>
<td>31.51</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>12.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>3.30</td>
<td>81.51</td>
<td>30.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.50</td>
<td>40.69</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1.73</td>
<td>91.77</td>
<td>73.86</td>
<td>1.01</td>
<td>0.1.11</td>
<td>51.01</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>11.1</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>8.8</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>4.31</td>
<td>34.02</td>
<td>51.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.50</td>
<td>40.79</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1.77</td>
<td>91.11</td>
<td>1.01</td>
<td>31.51</td>
<td>11.71</td>
<td>61.51</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>9.8</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>8.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>8.6</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>2.02</td>
<td>34.44</td>
<td>51.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CENTRAL LOW LIMIT</td>
<td>47.27</td>
<td>498.3</td>
<td>-746</td>
<td>658</td>
<td></td>
</tr>
<tr>
<td>3.66</td>
<td>37.31</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.96</td>
<td>55.96</td>
<td>751.01</td>
<td>42.21</td>
<td>11.41</td>
<td>41.72</td>
</tr>
<tr>
<td>1.0.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
<td>0.1.01</td>
</tr>
<tr>
<td>6.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>25.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>15.0</td>
<td>7.5</td>
<td>567</td>
<td>584</td>
<td></td>
</tr>
</tbody>
</table>

(continued)
<table>
<thead>
<tr>
<th>Value</th>
<th>0.52</th>
<th>0.52</th>
<th>0.50</th>
<th>0.80</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
<th>3.66</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.68</td>
<td>.61</td>
<td>.33</td>
<td>.391</td>
<td>.09</td>
<td>.961</td>
<td>.71</td>
<td>.51</td>
<td>.21</td>
<td>.61</td>
<td>.21</td>
<td>.11</td>
<td>.3</td>
<td>.96</td>
<td>.55</td>
<td>.55</td>
<td>.34</td>
<td>.20</td>
<td>.21</td>
</tr>
<tr>
<td></td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.91</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
</tr>
<tr>
<td>6.8</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>5.0</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.5</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.8</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
<tr>
<td>3.66</td>
<td>40.20</td>
<td>1</td>
<td>.75</td>
<td>.751</td>
<td>.034</td>
<td>.62</td>
<td>.481</td>
<td>.31</td>
<td>.11</td>
<td>.21</td>
<td>.01</td>
<td>.3</td>
<td>.821</td>
<td>.11</td>
<td>.61</td>
<td>.21</td>
<td>.00</td>
<td>.68</td>
<td>.68</td>
</tr>
<tr>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.10</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
</tr>
<tr>
<td>7.6</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>5.8</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>4.3</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.5</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
<tr>
<td>3.66</td>
<td>37.44</td>
<td>1</td>
<td>.141</td>
<td>.3</td>
<td>.68</td>
<td>.891</td>
<td>.4</td>
<td>.82</td>
<td>.961</td>
<td>.11</td>
<td>.51</td>
<td>.51</td>
<td>.51</td>
<td>.81</td>
<td>.21</td>
<td>.22</td>
<td>.01</td>
<td>.01</td>
<td>.92</td>
</tr>
<tr>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.10</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
<td>.90</td>
</tr>
<tr>
<td>8.3</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>5.6</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>4.3</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.3</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
<tr>
<td>3.66</td>
<td>43.19</td>
<td>1</td>
<td>.521</td>
<td>.2</td>
<td>.68</td>
<td>.89</td>
<td>.68</td>
<td>.75</td>
<td>.62</td>
<td>.11</td>
<td>.31</td>
<td>.41</td>
<td>.42</td>
<td>.11</td>
<td>.62</td>
<td>.01</td>
<td>.21</td>
<td>.41</td>
<td>.27</td>
</tr>
<tr>
<td>1.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
</tr>
<tr>
<td>6.7</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>5.3</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.8</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.1</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
<tr>
<td>3.66</td>
<td>39.66</td>
<td>1</td>
<td>.62</td>
<td>.68</td>
<td>.82</td>
<td>.55</td>
<td>.89</td>
<td>.98</td>
<td>.891</td>
<td>.31</td>
<td>.81</td>
<td>.61</td>
<td>.11</td>
<td>.62</td>
<td>.11</td>
<td>.81</td>
<td>.61</td>
<td>.7</td>
<td>.62</td>
</tr>
<tr>
<td>1.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
</tr>
<tr>
<td>7.1</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>6.9</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.5</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>4.1</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
<tr>
<td>3.66</td>
<td>36.60</td>
<td>1</td>
<td>.27</td>
<td>.32</td>
<td>.11</td>
<td>.3</td>
<td>.681</td>
<td>.21</td>
<td>.02</td>
<td>.32</td>
<td>.02</td>
<td>.52</td>
<td>.31</td>
<td>.81</td>
<td>.41</td>
<td>.1</td>
<td>.96</td>
<td>.20</td>
<td>.34</td>
</tr>
<tr>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>1.11</td>
<td>0.31</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
<td>.01</td>
</tr>
<tr>
<td>6.6</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>6.3</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>3.8</td>
<td>25.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
<td>4.1</td>
<td>15.0</td>
<td>7.5</td>
<td>.067</td>
<td>.81</td>
</tr>
</tbody>
</table>
Appendix B

PROGRAM FOR ESTIMATING THE RELATIONSHIP BETWEEN
CFS UNAVAILABILITIES AND NON-CFS UNAVAILABILITIES*

INTRODUCTION

Research conducted by students at UCLA on the allocation process
for the Los Angeles Police Department** found that unavailabilities
of patrol cars for reasons other than calls for service (due to traffic
enforcement, meal times, and the like) vary according to the cfs work-
load of the cars. These unavailabilities, which are an important com-
ponent of patrol car activity in all police departments, reduce the
effective number of cars available to service calls. Scheduled unavail-
abilities such as meals, and unscheduled but predictable activities such
as automobile service stops, can be expected to occur independent of the
call-for-service workload. However, discretionary officer-initiated
activity might increase during slack periods and decrease during over-
loaded periods of the day, or vice versa.

In Los Angeles, the variation was found to be rather complicated.
Data collected at the dispatch center showed that more time was spent
per car on non-cfs unavailabilities when the cfs workload was high than
when it was low. An apparent explanation for this is that the times of
day in which many emergencies are reported to the police by telephone
also have many activities visible from the street that require police
intervention.

However, the queuing delays experienced before a car could be dis-
patched to a call in Los Angeles indicated that more cars were unavail-
able at times of low cfs workload than were reported unavailable.
Moreover, it sometimes happened at times of high cfs workload that more
cars were actually available than were reported available. This is
because extra patrol units, such as traffic cars and sergeants' cars,
could be used to handle cfs work if absolutely necessary.

* This appendix was coauthored by David J. Jaquette, who also wrote
the program described here.

** An Analysis of the Patrol Car Deployment Methods of the Los
Angesles Police Department, Engineering School report by Public Systems
Analysis class, University of California at Los Angeles, 1975.
Since the equations in a patrol allocation program should be designed to predict queuing delays as they will actually occur, it is appropriate to estimate the effective number of patrol cars present in the field from data giving the number of calls delayed, and not from data telling how many cars were fielded and their reported unavailabilities.

Thus if NEFF denotes the effective number of patrol cars which, according to queuing formulas, would cause the observed fraction of calls delayed, our estimate of the fraction of time each car is unavailable on non-cfs activity is

$$UNAVL = 1 - \frac{NEFF}{CARS}, \quad (B.1)$$

where CARS is the number of cars fielded.

Then, the effective fraction of time unavailable (UNAVL) is modeled to be linearly related to the fraction of time the average car spends on calls for service, C:

$$UNAVL = B1 \cdot C + B2, \quad (B.2)$$

where B1 and B2 are coefficients specific to each precinct but assumed to be time homogeneous. This relationship was found to explain the relationship between effective and fielded cars in Los Angeles, as evidenced by the increase in the number of calls delayed during slack periods and the decrease during periods of heavy demand in calls for service. (See Fig. 3 in the User's Manual.)

Given this relationship, the only input data related to non-cfs unavailabilities needed by PCAM is the pair of unavailability parameters B1 and B2 for each precinct. The computer program listed and annotated here was written originally as an aid to the LAPD Automated Deployment of Available Manpower (ADAM) project in their attempt to implement a version of PCAM. It can be used to construct estimates of the unavailability parameters.
INPUT DATA

The program takes raw data which were available from LAPD records and converts them into numbers usable in a standard linear regression. Each data point read in on one data card represents a number of weeks (NWEKS) of aggregated data for one shift. For example, one line of printout in the data summary available to the LAPD would describe the activity of patrol cars in the Van Nuys Area during the tour from midnight to 3 a.m. on Mondays, over a four-week period; thus NWEKS = 4. Each input card contains the total number of actual car hours (AVLHR), which in this example would be 4x3 times the average number of cars fielded; the number of hours in the shift (NHOURS), which in the example is 3; actual hour spent on calls-for-service work (CFSWRK); total number of delayed calls (NDELAY); and total calls for service (NTCFS).

CALCULATIONS IN THE PROGRAM

The fraction of each car's time spent on calls for service, which is the independent variable of the regression, is immediately found as the calls-for-service workload divided by the total actual car-hours,

\[ C = \frac{CFSWRK}{AVLHR}. \]

This is calculated for each shift.

For each shift, dividing the number of calls delayed by the total number of calls gives the fraction delayed, which is an estimate of the probability of delay. If there are N effective cars on duty, and the number of cfs work hours per hour is \( \rho \), the formula for an M/M/N queue shows that the probability of a call being delayed is

\[ P(\text{delay}|N) = \frac{\rho^N}{(N!)^2(1-\rho/N)} \]  
\[ = \frac{\rho^N/(N!)(1-\rho/N)}{1 + \rho + \rho^2/2! + \cdots + \rho^{(N-1)}/(N-1)! + \rho^N/N!(1-\rho/N)}. \]  

(B.3)

A maximum-likelihood and unbiased estimate of the number of effective cars during a given shift can be made by solving \( N \) in the relationship

\[ P(\text{delay}|N) = \text{actual fraction of calls delayed}. \]
The value of $\rho$ needed in the above calculation is found as the actual call-for-service workload hours (CFSWRK) divided by the number of total hours contained in the data for that shift ($N\text{WEEKS} \times NHOURS$):

$$\rho = \frac{\text{CFSWRK}}{N\text{WEEKS} \cdot NHOURS}$$

Once $\rho$ and the fraction delayed are estimated, $N$ can be determined by evaluating the expression above for a $K$ such that

$$P(\text{delay}|K) > \text{actual fraction of calls delayed}$$

and

$$P(\text{delay}|K+1) < \text{actual fraction of calls delayed}.$$ 

Linear interpolation between $K$ and $K+1$ is used to estimate $N$. The ratio of $N$ to the actual number of cars fielded, CARS, gives an estimate of the fraction of time unavailable, UNAVL, as shown in Eq. B.1, ($\text{CARS} = \text{AVLHRS}/N\text{WEEKS} \cdot NHOURS$).

Once UNAVL and $C$ have been calculated for each shift in a precinct, the usual formulas for a regression fit are used to estimate $B1$ and $B2$ for that precinct:

$$B1 = \frac{n \sum \text{UNAVL}_i \cdot C_i - \sum \text{UNAVL}_i \sum C_i}{n \sum C_i^2 - (\sum C_i)^2},$$

where $n$ is the number of observations, and

$$B2 = \frac{\sum (\text{UNAVL}_i - B1 \cdot C_i)}{n}.$$
INPUT DATA FORMAT FOR PROGRAM TO CALCULATE B1 AND B2

The format instructions may be clarified by the sample data file that follows.

1. Control card. Enter the number of precincts for which data are provided in columns 1-2, format I2.

2. Cards for each precinct
   a. Precinct name. Enter precinct name on one card, left justified.
   b. Number of data cards for this precinct. Enter on one card in columns 1-2, format I2.
   c. Data cards. One for each shift.

<table>
<thead>
<tr>
<th>Position</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>F10.1</td>
<td>AVLHR: Number of actual car-hours fielded in the shift</td>
</tr>
<tr>
<td>11-12</td>
<td>I2</td>
<td>NHOURS: Number of hours in the shift</td>
</tr>
<tr>
<td>13-22</td>
<td>F10.2</td>
<td>CFSWRK: Number of car-hours of cfs work</td>
</tr>
<tr>
<td>23-25</td>
<td>I3</td>
<td>NDELAY: Number of calls delayed</td>
</tr>
<tr>
<td>26-28</td>
<td>I3</td>
<td>NTCFS: Total number of calls for service</td>
</tr>
</tbody>
</table>
SAMPLE DATA FILE FOR PROGRAM TO CALCULATE B1 AND B2

Note that an error has been purposely introduced for the first shift in WEST precinct. The number of calls delayed (64) exceeds the total number of calls (63). This data card (observation 1 in WEST precinct) will be ignored by the program.

<table>
<thead>
<tr>
<th>Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>WEST</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>111.</td>
</tr>
<tr>
<td>140.</td>
</tr>
<tr>
<td>312.</td>
</tr>
<tr>
<td>123.</td>
</tr>
<tr>
<td>240.</td>
</tr>
<tr>
<td>DOWNTOWN</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>105.</td>
</tr>
<tr>
<td>140.</td>
</tr>
<tr>
<td>336.</td>
</tr>
<tr>
<td>126.</td>
</tr>
<tr>
<td>245.</td>
</tr>
<tr>
<td>NORTH</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>122</td>
</tr>
<tr>
<td>152</td>
</tr>
<tr>
<td>362</td>
</tr>
<tr>
<td>138</td>
</tr>
<tr>
<td>210</td>
</tr>
<tr>
<td>128</td>
</tr>
<tr>
<td>158</td>
</tr>
</tbody>
</table>

OUTPUT FROM RUNNING PROGRAM WITH SAMPLE DATA FILE

<table>
<thead>
<tr>
<th>ERRIN</th>
<th>DELAY DATA FOR OBS</th>
<th>1 IN WEST</th>
<th>PRECINCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOR WEST</td>
<td>PRECINCT</td>
<td>B1 = -0.3476</td>
<td>B2 = 0.5876</td>
</tr>
<tr>
<td>FOR DOWNTOWN</td>
<td>PRECINCT</td>
<td>B1 = -0.7170</td>
<td>B2 = 0.7476</td>
</tr>
<tr>
<td>FOR NORTH</td>
<td>PRECINCT</td>
<td>B1 = -0.6336</td>
<td>B2 = 0.7015</td>
</tr>
</tbody>
</table>
LISTING OF PROGRAM TO CALCULATE UNAVAILABILITY
PARAMETERS B1 AND B2
DIMENSION PROB(20)
INTEGER FACTN(4)
DATA NWEKS/4/

C C CALCULATE FACTORIAL(I) AS FACTN(I+1)
C
FACTN(1) = 1,
DO 1 I=2,40
    FACTN(I) = FACTN(I-1)*I
1 CONTINUE

C READ (5,101) NODIST
101 FORMAT(I2)
DC 30 IJ=1,NODIST
READ (5,102) PCTNM1,PCTNM2,PCTNM3
102 FORMAT(3A4)
READ (5,103) NOBSV
103 FORMAT(I2)
SUMY=0.0
SUMYSQ=0.0
SUMC=0.1
SUMCSQ=0.0
SUMYC=0.0
NOBS = 0
DC 20 IX=1,NOBSV
READ (5,104) AVLHP,NHCRS,CFSWRK,NDELAY,NTCFS
CARS = AVLHP/(NWEKS*NHCRS)
NCARS = CARS +.99999999
104 FORMAT(F10.1,112,F10.2,213)
RHO= CFSWRK/(NHCRS*NWEKS)
DELAYP = NDELAY
DELAYP = DELAYP/NTCFS
IF (DELAYP.GT.1.0) GO TO 19

C C CALCULATE INTEGER N-EFFECTIVE FROM QUEUING FORMULA
C
NEFF = 1
LOWCAR = RHO + 1
DO 5 I=LOWCAR,NCARS
    DENSUM = I
    ILFSS1 = I-1
    DO 4 I=1,ILFSS1
        DENSUM = DENSUM * RHO**I + FACTN(I+1)
        XNUM = RHO**I*(1.-RHO/I)**FACTN(I+1)
        PROB(I) = XNUM/(DENSUM+XNUM)
        NEFF = I
        IF (PROB(I).LE.DELAYP) GO TO 11
3 CONTINUE

5 CONTINUE

C C THE FOLLOWING IS AN INTERPOLATION FOR EFFECTIVE N
C IF THE CLOSEST N-EFFECTIVE CARS IS GREATER THAN OR EQUAL TO THE
C ACTUAL NUMBER OF CARS, THEN THE INTERPOLATION IS BYPASSED, AND
C THE TIME SPENT ON NON-CF WORK IS SET TO ZERO
C
AEFF = NEFF
IF (AEFF.GE.CARS) GO TO 7
JK = NEFF - 1
IF (NEFF.GT.LOWCAP) AEFF = PROB(NEFF)-DELAYP/
1  (PROB(JK)-PROB(NEFF)) + AEFF
   IF (NEFF.EQ.0.0) EFFN=(1.0-DELAYP)/
1  (1.0-PROB(NEFF)) + RHO
   UNAVL = AMAX1(0.0,1-EFFN/CARS)
   GO TO 8
   UNAVL = 0.0
   EFFN = CARS

ACCUMULATE TERMS FOR REGRESSION COEFFICIENTS

   C = RHO/CARS
   SUMY = SUMY + UNAVL
   SUMYSQ = SUMYSQ + UNAVL*UNAVL
   SUMC = SUMC + C
   SUMCSQ = SUMCSQ + C*C
   SUMYC = SUMYC + UNAVL*C
   NOBS = NOBS+1
   GO TO 20

WRITE(6,123) JK,PCTNM1,PCTNM2,PCTNM3
   FORMAT(' ERRCD IN DELAY DATA FOR OBS',I4,
1      ' IN ',3A4,' PRECINCT')
   CONTINUE

CALCULATE REGRESSION COEFFICIENTS

   YC = NOBS*SUMYC-SUMY*SUMC
   CC = NOBS*SUMCSQ-SUMC*SUMC
   B1 = YC/CC
   B2 = SUMY/NOBS - B1*SUMC/NOBS

WRITE(6,106) PCTNM1,PCTNM2,PCTNM3,B1,B2
   FORMAT('0 FOR ',3A4,' PRECINCT B1=',F10.4,' B2=',
1      F10.4)
   CONTINUE

CALL EXIT
END
### Appendix C

**PROGRAM CROSS-REFERENCE TABLE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Defined in</th>
<th>Referenced in</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDALC</td>
<td>ADDALC</td>
<td>MAIN</td>
</tr>
<tr>
<td>ADDCAR</td>
<td>ADDCAR</td>
<td>ADDALC</td>
</tr>
<tr>
<td>ADJUST</td>
<td>ADJUST</td>
<td>ADDALC</td>
</tr>
<tr>
<td>AVTT</td>
<td>AVTT</td>
<td>COMPTB,KNSTR</td>
</tr>
<tr>
<td>CEIL</td>
<td>CEIL</td>
<td>DERIVE,MEET,ADDALC</td>
</tr>
<tr>
<td>CKOVR</td>
<td>CKOVR</td>
<td>MEET,ADDALC,WRITE</td>
</tr>
<tr>
<td>COMPTB</td>
<td>COMPTB</td>
<td>DSPDTP,DSPDTP</td>
</tr>
<tr>
<td>DERIVE</td>
<td>DERIVE</td>
<td>READ,SET</td>
</tr>
<tr>
<td>DISP</td>
<td>DISP</td>
<td>MAIN</td>
</tr>
<tr>
<td>DSPDTP</td>
<td>DSPDTP</td>
<td>DSP</td>
</tr>
<tr>
<td>DSPDPT</td>
<td>DSPDPT</td>
<td>DSP</td>
</tr>
<tr>
<td>GETBOT</td>
<td>GETBOT</td>
<td>INIT,READ</td>
</tr>
<tr>
<td>GETKNN</td>
<td>GETKNN</td>
<td>SCAN</td>
</tr>
<tr>
<td>GETTOP</td>
<td>GETTOP</td>
<td>SCAN,ADDALC,WRITE</td>
</tr>
<tr>
<td>GTDSPC</td>
<td>GTDSPC</td>
<td>READ,LIST,DISP,SET,MEET,ADDALC,WRITE</td>
</tr>
<tr>
<td>INIT</td>
<td>INIT</td>
<td>MAIN</td>
</tr>
<tr>
<td>KEYWDS</td>
<td>DATA</td>
<td>MAIN,INIT,SCAN,READ,GTDSPC,DERIVE,LIST,SETWFL,DISP,DSPDTP,DSPDTP,SET,MEET,ADDALC,WRITE</td>
</tr>
<tr>
<td>KNSTR</td>
<td>KNSTR</td>
<td>MEET</td>
</tr>
<tr>
<td>LCODES</td>
<td>DATA</td>
<td>SCAN,GETKNN</td>
</tr>
<tr>
<td>LIST</td>
<td>LIST</td>
<td>MAIN</td>
</tr>
<tr>
<td>LKP1</td>
<td>LKP1</td>
<td>GETKNN,READ,GTDSPC,MRGORD</td>
</tr>
<tr>
<td>LKP8</td>
<td>LKP8</td>
<td>SCAN,READ,SETWFL,NXPCT</td>
</tr>
<tr>
<td>MEET</td>
<td>MEET</td>
<td>MAIN</td>
</tr>
<tr>
<td>MOVE</td>
<td>MOVE</td>
<td>SCAN,READ,MRGORD,WRITE</td>
</tr>
<tr>
<td>MRGORD</td>
<td>MRGORD</td>
<td>READ,DISP</td>
</tr>
<tr>
<td>NXDAY</td>
<td>NXDAY</td>
<td>LIST,DSPDTP,DSPDTP,SET,MEET,ADDALC,ADDCAR,WRITE</td>
</tr>
<tr>
<td>NXPCT</td>
<td>NXPCT</td>
<td>LIST,DSPDTP,DSPDTP,SET,MEET,ADDALC,ADDCAR,WRITE</td>
</tr>
<tr>
<td>NXTOUR</td>
<td>NXTOUR</td>
<td>LIST,DSPDTP,DSPDTP,SET,MEET,STRCAR,ADDALC,ADDCAR,WRITE</td>
</tr>
<tr>
<td>OBJFUN</td>
<td>OBJFUN</td>
<td>SBLOBJ,ADJUST</td>
</tr>
<tr>
<td>OBJF1</td>
<td>OBJF1</td>
<td>COMPTB,KNSTR,OBJFUN</td>
</tr>
<tr>
<td>OBJF2</td>
<td>OBJF2</td>
<td>COMPTB,KNSTR,OBJFUN</td>
</tr>
<tr>
<td>OBJF3</td>
<td>OBJF3</td>
<td>COMPTB,KNSTR,OBJFUN</td>
</tr>
<tr>
<td>OFFSET</td>
<td>DATA</td>
<td>READ,DERIVE,SBACT,SBLEF,LIST,NXPCT,NXDAY,NXTOUR,DSPDTP,DSPDTP,COMPTB,AVTT,OBJF2,OBJF3,SET,MEET,STRCAR,KNSTR,ADDALC,SBLOBJ,STRDF,ADJUST,STROBJ,OBJFUN,ADDCAR,WRITE</td>
</tr>
<tr>
<td>PNTRS</td>
<td>DATA</td>
<td>INIT,READ,GTDSPC,DERIVE,SBACT,SBLEF,LIST,SETWFL,NXPCT,NXDAY,NXTOUR,DSPDTP,DSPDTP,COMPTB,SET,MEET,STRCAR,KNSTR,CKOVR,ADDALC,SBLOBJ,STRDF,ADJUST,STROBJ,OBJFUN,ADDCAR,WRITE</td>
</tr>
<tr>
<td>PQQUEUE</td>
<td>PQQUEUE</td>
<td>OBJF1,OBJF2,OBJF3</td>
</tr>
<tr>
<td>Symbol</td>
<td>Defined in</td>
<td>Referenced in</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>PRTL</td>
<td>PRTL</td>
<td>DSPDPT, DSPDT, TOTAL</td>
</tr>
<tr>
<td>READ</td>
<td>READ</td>
<td>MAIN</td>
</tr>
<tr>
<td>SBLACT</td>
<td>SBLACT</td>
<td>DERIVE, MEET, ADDALC</td>
</tr>
<tr>
<td>SBLEF</td>
<td>SBLEF</td>
<td>DERIVE, MEET, ADDALC</td>
</tr>
<tr>
<td>SBLOBJ</td>
<td>SBLOBJ</td>
<td>ADDALC, ADJUST, ADDCAR</td>
</tr>
<tr>
<td>SCAN</td>
<td>SCAN</td>
<td>MAIN, READ, GTDSPC, LIST, DISP, SET, MEET, ADDALC, WRITE</td>
</tr>
<tr>
<td>SCODES</td>
<td>DATA</td>
<td>MAIN, SCAN, READ, GTDSPC, LIST, DISP, SET, MEET, ADDALC, WRITE</td>
</tr>
<tr>
<td>SET</td>
<td>SET</td>
<td>MAIN</td>
</tr>
<tr>
<td>SETWFL</td>
<td>SETWFL</td>
<td>LIST, DISP, SET, MEET, ADDALC, WRITE</td>
</tr>
<tr>
<td>SKIP</td>
<td>SKIP</td>
<td>READ</td>
</tr>
<tr>
<td>STATS</td>
<td>DATA</td>
<td>READ, DISP, ZERO, COMPTB, PRTL, TOTAL</td>
</tr>
<tr>
<td>STORE</td>
<td>DATA</td>
<td>MAIN, INIT, GETBOT, SCAN, GETTOP, READ, DERIVE, SBLACT, SBLEF, LIST, SETWFL, NXPCT, NXDAY, NXTOUR, DISP, DSPDPT, DSPDT, COMPB, AVTT, OBJF1, OBJF2, OBJF3, SET, MEET, STRCAR, KINST, CKOV, ADDALC, SBLOBJ, STRDF, ADJUST, STROBJ, OBJFUN, ADDCAR, WRITE</td>
</tr>
<tr>
<td>STRCAR</td>
<td>STRCAR</td>
<td>MEET, ADDALC</td>
</tr>
<tr>
<td>STRDF</td>
<td>STRDF</td>
<td>STROBJ, ADDCAR</td>
</tr>
<tr>
<td>STROBJ</td>
<td>STROBJ</td>
<td>ADDALC, ADJUST, ADDCAR</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>DATA</td>
<td>MAIN, INIT, GETBOT, SCAN, GETTKN, GETTOP, READ, GTDSPC, DERIVE, LIST, SETWFL, DISP, DSPDPT, DSPDT, TITLE, PRTL, TOTAL, OBJF2, SET, MEET, CKOV, ADDALC, ADDCAR, WRITE</td>
</tr>
<tr>
<td>TITLE</td>
<td>TITLE</td>
<td>DSPDPT, DSPDT</td>
</tr>
<tr>
<td>TOTAL</td>
<td>TOTAL</td>
<td>DSPDPT, DSPDT</td>
</tr>
<tr>
<td>WRITE</td>
<td>WRITE</td>
<td>MAIN</td>
</tr>
<tr>
<td>ZERO</td>
<td>ZERO</td>
<td>DSPDPT, DSPDT</td>
</tr>
</tbody>
</table>
Appendix D

ADDRESSES FOR FURTHER INFORMATION

1. For copies of the PCAM program on card or tape, answers to questions about the program, and information about related emergency service deployment models:

Dr. Jan M. Chaiken
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

2. Research sponsors:

U.S. Department of Housing and Urban Development

Alan Siegel, Director
Hartley Campbell Pitts, Program Manager

Office of Policy Development and Research
Community Development and Management
Research Division
451 Seventh Street, S.W.
Washington, D.C. 20410
(202) 755-6970

National Institute of Law Enforcement and Criminal Justice

Dr. Richard Linster
Law Enforcement Assistance Administration
National Institute of Law Enforcement and Criminal Justice
Office of Evaluation
633 Indiana Avenue, N.W.
Washington, D.C. 20530
(202) 376-3933